SMART CAMPUS SECURITY: FACIAL RECOGNITION – BASED ENTRY AND ATTENDANCE MONITORING SYSTEM

A Thesis Presented
to the Undergraduate School
HGBaquiran College
Centro, Tumauini, Isabela

In Partial Fulfillment
of the Requirements for the Degree
BACHELOR OF SCIENCE IN
INFORMATION TECHNOLOGY

By

DIAZ, JESUS D.
MANALO, CHELLSY FEI B.
ONG, DAN ISENLOIS L.
SAGADRACA, CRISTINE JOY Q.

FEBRUARY 2025

APPROVAL SHEET

In partial fulfillment of the requirements for the degree, BACHELOR OF SCIENCE IN INFORMATION TECHOLOGY, this study titled, **"SMART CAMPUS SECURITY: FACIAL RECOGNITION – BASED ENTRY AND ATTENDANCE MONITORING SYSTEM",** has been prepared and submitted by **CRISTINE JOY Q. SAGADRACA, JESUS D. DIAZ, CHELLSY FEI B. MANALO,** and **DAN ISENLOIS L. ONG** who is hereby recommended for Oral Examination.

RAFFY A. MELEGRITO
Adviser

Approved by the Panel for Oral Examination with a grade of PASSED.

PANEL OF EXAMINERS

JEREMIAH C. LAMAN
Member

KIM S. TARAYAO, LPT
Member

Accepted and approved in partial fulfillment of the requirements for the degree, **BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY.**

RAFFY A. MELEGRITO
Program Coordinator, BSIT

ACKNOWLEDGMENT

.

The successful completion of this thesis, "Smart Campus Security: Facial Recognition – Based Entry and Attendance Monitoring System," represents the culmination of significant effort and wouldn't have been possible without the support and guidance of numerous individuals and entities.

First and foremost, We extend our deepest appreciation to our thesis adviser, Sir Raffy A. Melegrito, for your invaluable guidance, support and encouragement, and to the panel of examiners, Sir Kim S. Tarayao, LPT and Sir Jeremiah C. Laman, for your constructive feedback, criticism and suggestions. We also appreciate the support from the faculty and staff of the Information Technology Department.

To the researchers Cristine Joy Q. Sagadraca, Jesus D. Diaz, Chellsy Fei B. Manalo, Dan Isenlois L. Ong, Thank you for the collaboration, efforts and dedication. And also we are grateful to the students who participated in this study and your willingness to share your experiences and valuable insights.

Also we deeply grateful to our family, friends and contributors for your unwavering support and encouragement throughout our academic journey. This research would not have been possible without the collective efforts of all these remarkable individuals. Thank you for being an integral part of this journey and for helping us to reach this important milestone.

The Researcher

DEDICATION

This study is dedicated to our families, whose unwavering love and support have been the foundation of our academic journey. Your encouragement has inspired us to pursue our dreams, and we are deeply grateful for your sacrifices and belief in us.

We also dedicate this work to our teachers and mentors, whose wisdom and dedication to education have ignited our passion for learning and shaped our understanding of the world. Thank you for your invaluable insights.

To our fellow students, your camaraderie has made this journey memorable. Together, we have navigated challenges and celebrated successes, cherishing the friendships we've built.

And lastly, we dedicate this study to future Information Technology students, hoping it serves as a stepping stone for your academic and professional success in computer literacy. May it inspire you to pursue knowledge and continuous improvement.

The Researcher

ABSTRACT

This study, "Smart Campus Security: Facial Recognition-Based Entry and Attendance Monitoring System," develops and evaluates a system to automate security and attendance tracking in educational institutions. In modern campuses, efficient and secure monitoring is crucial for safety and operational effectiveness.

We employed an Agile methodology and ISO/IEC 25010 standards to guide the development and assess the system. The system includes modules for student registration, facial recognition, real-time monitoring, and administrative management. Data was gathered through alpha and beta testing, and user surveys.

Results indicated "Excellent" usability and "Good" performance in core functions like reliability and functional suitability. The system demonstrated adequate operational and structural qualities in performance efficiency, security, and maintainability. Compatibility and portability were identified as areas for optimization

These findings highlight the potential of facial recognition technology to enhance campus security and streamline attendance processes. We recommend optimizing system performance and expanding compatibility for future development. This research provides a foundation for creating smarter, safer, and more efficient learning environments. Investing in such systems can transform campus management and improve security protocols.

TABLE OF CONTENTS

CONTENT **PAGE**

TITLE
PAGE i
APPROVAL SHEET.....ii
ACKNOWLEDGMENT.....iii
DEDICATION.....iv
ABSTRACT.....v
TABLE OF CONTENTS.....vi
LIST OF TABLES.....viii
LIST OF FIGURES.....x

CHAPTER I. INTRODUCTION

Background and Problem of the Study.....1
Objectives of the Study.....3
Scope of the Study.....5
Limitations of the Study.....6
Significance of the Study.....7
Review of Related Literature and Studies.....8
Synthesis of the Review.....10
Definition of Terms.....10

CHAPTER II. TECHNICAL BACKGROUND

Conceptual Framework.....13
System Architecture.....14
Context Diagram.....15
Data Flow Diagram.....15
Entity Relationship Diagram.....17
Database Schema.....18
Gantt Chart.....20

CHAPTER III. METHODOLOGY

Software Development Methodology.....27
Phases of Agile Development Methodology.....28
Data Gathering Techniques.....28
Development Requirements.....29
Respondents of the Study.....30
Sampling Technique.....30
Statistical Treatment.....31

CHAPTER IV. RESULTS AND DISCUSSION

Results, Analysis, and Interpretation of Data.....32

CHAPTER V. SUMMARY, CONCLUSION, AND RECOMMENDATIONS

Summary.....44

Conclusion.....45

Recommendations.....45

BIBLIOGRAPHY.....48
APPENDICES.....50
CURRICULUM VITAE.....125

LIST OF TABLES

Table	Page
Table 1: User Table Schema	18
Table 2: Attendace Table Schema	19
Table 3: Access Table Schema	19
Table 4: Notification Table Schema	20
Table 5: Activities for the Month of February	21
Table 6: Activities for the Month of March	21
Table 7: Activities for the Month of April	22
Table 8: Activities for the Month of May	22
Table 9: Activities for the Month of June	23
Table 10: Activities for the Month of July	23
Table 11: Activities for the Month of August	23
Table 12: Activities for the Month of September	24
Table 13: Activities for the Month of October	24
Table 14: Activities for the Month of November	24
Table 15: Activities for the Month of December	25
Table 16: Activities for the Month of January	25
Table 17: Activities for the Month of February	26
Table 18: Activities for the Month of February	26
Table 19: Hardware Requirements	29
Table 20: Software Requirements	29
Table 21: Stratified Respondent Categories	30

Table 22: Likert Scale 31

Table 23: Interpretation 31

Table 24: Functional Suitability 39

Table 25: Performance Efficiency 40

Table 26: Portability 40

Table 27: Compatibility 41

Table 28: Usability 41

Table 29: Reliability 42

Table 30: Security 42

Table 31: Maintainability 43

LIST OF FIGURES

Figure	Page
Figure 1: Conceptual Framework	13
Figure 2: System Architecture	14
Figure 3: Context Diagram	15
Figure 4: Data Flow Diagram	16
Figure 5: Entity Relationship Diagram	17
Figure 6: Agile Software Development Methodology	27
Figure 7: Main Application Menu	32
Figure 8: System Settings	33
Figure 9: Facial Recognition Configuration	34
Figure 10: Account Permission	34
Figure 11: Student Registration	35
Figure 12: Monitoring Module	36
Figure 13: Alpha Testing	37
Figure 14: Beta Testing	38

CHAPTER I

THE PROBLEM AND ITS BACKGROUND

The evolution of attendance monitoring systems had transitioned from traditional manual methods to more advanced automated solutions. Educational institutions had faced persistent challenges in ensuring accurate, secure, and efficient attendance tracking. While manual methods and early automation technologies such as Radio-Frequency Identification (RFID) and biometric authentication had been widely implemented, each approach presented significant limitations that hindered their reliability and effectiveness.

Manual attendance tracking, which relied on paper-based logs or manual input into digital records, was the most time-consuming and error-prone method. This system was highly susceptible to human errors, such as mis recording, omission of entries, and inconsistencies in data entry (Tan, 2018). Additionally, it facilitated attendance fraud, including proxy attendance, where students signed in for absent peers, undermining the integrity of the system (Pramod Kumar et al., 2023). The administrative burden of manually verifying and consolidating attendance records further complicated institutional management, reducing efficiency and increasing workload for educators and staff (El-Mashad & Ali, 2024).

RFID technology introduced automation in attendance tracking by allowing students to use ID cards embedded with RFID tags to register their presence. However, despite its improvements over manual tracking, RFID-based systems still suffered from several challenges. Security vulnerabilities remained a major issue, as RFID cards could be lost, stolen, or duplicated, leading to potential identity fraud and unauthorized access (Thaleeparambil et al., 2024). The reliance on physical cards also meant that students could engage in buddy punching, where one student scanned another’s card to falsify attendance (Seah, 2023). Moreover, RFID

systems required significant investment in scanners, cards, and maintenance, which could be financially burdensome (Rajesh et al., 2024). Operationally, RFID scanners necessitated physical scanning, which could lead to bottlenecks and inefficiencies, particularly in large classrooms or at entry points (Bourdima, 2024).

Biometric authentication, particularly fingerprint-based systems, offered greater security than RFID by using unique biological traits for identity verification. However, it was not without its challenges. Biometric attendance systems required physical contact, making them prone to hygiene concerns, particularly in the post-COVID-19 era (Lateef & Kamil, 2023). Additionally, fingerprint scanners could fail to recognize users due to factors like skin dryness, dirt, or injuries, leading to inaccuracies (Karthick et al., 2024). Unlike facial recognition, which allowed multiple users to be authenticated simultaneously, biometric fingerprint systems required individual processing, which could cause delays, particularly in large student populations (Zainuddin et al., 2024). Moreover, biometric data storage raised privacy concerns, and the need for specialized hardware and maintenance added to institutional costs (Gudikandhula et al., 2023).

To overcome these challenges, modern attendance monitoring systems had increasingly adopted facial recognition technology due to its contactless, secure, and efficient nature. Unlike RFID and biometric fingerprint systems, facial recognition could authenticate multiple individuals in real time without requiring physical interaction (Pramod Kumar et al., 2023). Furthermore, AI-driven facial recognition enhanced accuracy, security, and integration with cloud-based platforms, reducing administrative workload while improving institutional security (El-Mashad & Ali, 2024).

In response to these technological advancements, this study aimed to develop a Smart Campus Security System that integrated facial recognition technology for automated entry authentication

and attendance tracking. This initiative sought to modernize attendance monitoring, minimize human intervention, enhance security, and streamline administrative workflows, ultimately transforming campus management into a more intelligent and data-driven ecosystem.

Objectives of the Study

1. General Objective

The primary objective of this study was to develop a Smart Campus Security System that utilized facial recognition technology for automated entry authentication and attendance monitoring in educational institutions. The system aimed to provide a contactless, efficient, and secure attendance tracking solution while enhancing campus security through real-time verification and access control.

2. Specific Objectives

- To design and develop a facial recognition-based attendance monitoring system that automated students attendance tracking.
- To integrate the system with cloud-based storage and real-time access control to enable centralized management of attendance records.
- To test and evaluate the system based on ISO/IEC 25010 software quality standards to ensure performance, usability, and security compliance.

3. Identified Problems

The study sought to address the following challenges:

- Inefficiencies in manual attendance tracking, including errors, time consumption, and administrative burden.

- Security vulnerabilities in RFID-based systems, such as lost or duplicated ID cards and proxy attendance.
- Hygiene and usability concerns with biometric fingerprint systems, which required physical contact and had accuracy issues.
- The need for a contactless and real-time authentication system to improve security and operational efficiency in educational institutions.

4. System Development Modules and Submodules

The proposed system was consisted of the following key modules and submodules:

- User Management Module
  - Registration of students, faculty, and administrators
  - Role-based access control
  - Facial data enrollment and verification
- Facial Recognition Module
  - Image processing and feature extraction
- Attendance Monitoring Module
  - Real-time attendance logging and tracking
  - Automated attendance report generation
- Security & Access Control Module
  - Entry authentication based on facial recognition
  - Unauthorized access detection and alert system
- Database and Cloud Integration Module
  - Cloud-based storage of attendance and access logs
  - Data encryption and security protocols
  - Backup and recovery features

5. Testing Approach

The system underwent a two-stage testing process—Alpha and Beta testing—to ensure its technical robustness and practical usability before full deployment.

- **Alpha Testing:** This initial phase was conducted by the development team and a small group of IT experts in a controlled lab environment. The focus was on verifying the core functionality of all modules, including the accuracy of the facial recognition algorithm under different lighting conditions, the reliability of the attendance logging system, the security of the access control module, and the integrity of database operations. Unit and integration tests were performed to identify and resolve bugs.
- **Beta Testing (User Acceptance Testing - UAT):** Following successful Alpha testing, the system was deployed to a selected group of end-users (students, faculty, administrators) in a real-world campus setting. This phase evaluated the system's usability, performance efficiency under load, and overall effectiveness in streamlining the attendance and entry process. Feedback on the user interface, recognition speed, and accuracy was collected and used to make final refinements.

6. Evaluation Approach (ISO/IEC 25010)

The system was evaluated using ISO/IEC 25010 software quality standards, focusing on the following criteria:

- Functional Suitability
- Performance Efficiency
- Usability
- Reliability
- Security
- Maintainability
- Portability

Scope of the Study

This study focused on the development and implementation of a Smart Campus Security System that utilized facial recognition technology for automated entry authentication and attendance monitoring in educational institutions. The system was designed to enhance security, accuracy, and efficiency in attendance tracking while mitigating the risks associated with manual, RFID, and biometric-based systems.

The system included essential modules for user management (registration, role-based access control, and facial data enrollment), real-time facial recognition (image processing and feature extraction for authentication), automated attendance tracking (logging and report generation), and security (unauthorized access detection with alerts). Data were securely stored in a cloud-based database with encryption and backup protocols. The project deliberately excluded advanced analytics, mobile app integration, multi-factor authentication, and offline operation to maintain simplicity. Technical boundaries included using standard webcams, pre-trained AI models, and free-tier cloud storage. Key stakeholders were students (check-in via facial scan), faculty (attendance reports), and administrators (user management and security monitoring). The proposed system was a desktop application.

Limitations of the Study

While the system aimed to improve campus security and attendance monitoring, there were certain limitations that were beyond the scope of this study:

1. **Limited to Facial Recognition-Based Authentication**

- The system did not support other authentication methods such as fingerprint scanning, RFID, or QR code-based verification.
- Individuals who had facial obstructions (e.g., masks, sunglasses, or poor lighting conditions) might have experienced recognition inaccuracies.

2. **Dependence on Camera Quality and Network Connectivity**
   - The effectiveness of facial recognition was dependent on high-resolution cameras and proper lighting conditions.
   - It required stable internet connectivity for real-time data synchronization with cloud servers.

3. **Privacy and Ethical Concerns**
   - While data encryption and access control measures were implemented, privacy concerns related to biometric data storage and usage remained a challenge.
   - The study did not cover legal and regulatory compliance for biometric data protection laws across different regions.

4. **Hardware and Infrastructure Requirements**
   - The system required dedicated hardware, including high-definition cameras and cloud-based servers, which might have been not feasible for all institutions due to budget constraints.
   - It did not support offline mode; real-time processing required continuous power supply and internet access.

5. **Limited to Educational Institutions**

- The study was specifically designed for schools, colleges, and universities and did not cover corporate offices, government facilities, or other organizations.
- The system was not designed for large-scale public surveillance or law enforcement applications.

Significance of the Study

This study would benefited multiple stakeholders by improving security, efficiency, and accuracy in attendance tracking within educational institutions. The beneficiaries and their corresponding advantages were as follows:

Administrators – Gained real-time monitoring, automated reporting, and reduced manual workload.

Educational Institutions – Enhanced campus security, ensured accurate attendance records, and improved operational efficiency.

Faculty Members – Saved time with automated roll-calling and gained instant access to attendance data.

IT Personnel & System Developers – Acquired practical experience in facial recognition and security system integration.

Parents & Guardians – Monitored student attendance in real-time, ensuring safety and accountability.

Researchers & Future Developers – Utilized the study as a reference for advancing biometric and AI-based security solutions.

Security Personnel – Benefited from automated identity verification and real-time alerts for unauthorized access.

Students – Experienced a seamless, contactless attendance process while reducing fraudulent check-ins.

Review of Related Literature and Studies (RRL)

The development of a Smart Campus Security: Facial Recognition-Based Entry and Attendance Monitoring System was rooted in the evolution of attendance tracking and security technologies. This review explored existing literature thematically, covering manual attendance challenges, RFID and biometric-based approaches, facial recognition systems, and software development methodologies.

Challenges of Traditional Attendance Systems

Manual attendance tracking remained time-consuming, error-prone, and susceptible to fraud such as proxy attendance. Tan (2018) highlighted that manual methods required extensive human intervention, leading to inefficiencies in large institutions. Moreover, paper-based logs posed security risks, as they could be easily manipulated or lost. The need for an automated and tamper-proof solution had driven advancements in attendance tracking systems.

RFID-Based Attendance Systems

RFID technology was one of the earliest automation attempts, using card-based identification for attendance logging (Thaleeparambil et al., 2024). While RFID reduced manual effort, it suffered from high costs, maintenance issues, and security risks due to the possibility of lost, stolen, or cloned cards. Similarly, Gudikandhula et al. (2023) pointed out that RFID systems required physical interaction, making them less efficient in ensuring seamless attendance tracking.

Biometric-Based Attendance Systems

Biometric authentication, particularly fingerprint scanning, provided a more secure and unique identification method. However, Pramod Kumar et al. (2023) and El-Mashad & Ali (2024) noted several challenges, including long processing times, high costs, and hygiene concerns, particularly in the post-pandemic era. Biometric

systems also faced false acceptance and rejection rates, affecting accuracy and user experience.

Facial Recognition as an Advanced Solution

Facial recognition has emerged as a contactless, efficient, and accurate attendance tracking method (Seah, 2023; Karthick et al., 2024). Unlike RFID and fingerprint systems, facial recognition allowed for simultaneous identification of multiple individuals, reducing human intervention and minimizing attendance fraud (Lateef & Kamil, 2023). Additionally, studies by Thirukrishna et al. (2023) and Rajesh et al. (2024) emphasized the real-time processing capabilities of facial recognition, making it ideal for security applications in educational institutions.

Enhancing Campus Security with Integrated Technologies

Advanced attendance systems integrated Wi-Fi, IoT, and AI-driven recognition technologies to enhance accuracy and security (Thaleeparambil et al., 2024; Zainuddin et al., 2024). Bourdima (2024) demonstrated how real-time alerts and automated security protocols could be implemented to restrict unauthorized campus access. Furthermore, Patel et al. (2018) explored the combination of IoT and facial recognition for attendance tracking, emphasizing its scalability and adaptability in smart campus environments.

Software Development Model

The system followed an Agile Software Development Model, ensuring flexibility, iterative improvements, and continuous feedback throughout development. Agile methodologies were particularly useful for AI-based systems, allowing developers to fine-tune facial recognition accuracy and enhance security features based on testing and user feedback. The integration of ISO 25010 standards ensured the system's functionality, performance, security, usability, and reliability.

Synthesis of the Review

Existing studies had demonstrated the limitations of manual, RFID, and biometric-based attendance systems, highlighting the advantages of facial recognition for contactless, secure, and efficient monitoring. The integration of IoT, AI, and cloud-based systems further enhanced real-time security and scalability. This study built upon these technologies to develop a comprehensive, AI-driven facial recognition system for campus security and attendance tracking, addressing previous limitations and ensuring a smarter, safer, and more efficient learning environment.

Definition of Terms

The following terms were defined according to their relevance and usage within this study:

Artificial Intelligence (AI) – The simulation of human intelligence by machines, used in this study to enhance the accuracy and efficiency of facial recognition in attendance monitoring.

Automated Attendance System (AAS) – A system that recorded and verified attendance without manual intervention, utilizing facial recognition technology for efficiency and security.

Biometric Authentication – A security process that verified an individual's identity using unique physical characteristics, such as fingerprints or facial features, which was essential in the proposed system.

Cloud-Based System – A system that stored and processed data over the internet instead of local servers, used for real-time attendance tracking and security monitoring.

Deep Learning (DL) – A subset of AI that utilized neural networks to improve facial recognition accuracy and enhance system performance.

Face Detection – The process of identifying human faces in images or video frames, which was the first step in facial recognition attendance tracking.

Face Recognition Technology (FRT) – An AI-driven process that matched a detected face against a stored database to verify identity, serving as the core technology in this system.

Global Positioning System (GPS) – A satellite-based navigation system used to track the location of students during attendance verification for enhanced security.

Internet of Things (IoT*)* – A network of connected devices that communicated over the internet, used in the system to integrate facial recognition with campus security features.

ISO 25010 – An international software quality standard used in this study to evaluate the system's performance, security, and usability.

Machine Learning (ML) – A branch of AI that enabled the system to improve its facial recognition capabilities over time by analyzing patterns and user data.

Manual Attendance System (MAS) – The traditional method of taking attendance using pen-and-paper or manual logging, which this study aimed to replace.

Near-Field Communication (NFC) – A short-range wireless technology used in RFID-based attendance systems, which was compared to facial recognition in this study.

Radio Frequency Identification (RFID) – A technology that uses electromagnetic fields to identify and track attendance through RFID cards, discussed as an alternative in this study.

Smart Campus Security System (SCSS) – The integration of AI, IoT, and facial recognition technology to enhance security and attendance monitoring in educational institutions.

Software Development Life Cycle (SDLC) – The process of designing, developing, and maintaining software, followed in the creation of this system.

Spoofing Attack – A fraudulent attempt to trick a facial recognition system using fake images or videos, which the proposed system aimed to prevent through AI-driven security features.

Wi-Fi-Based Authentication (WBA) – A method that used Wi-Fi signals to verify the presence of a user, which was considered as a supporting feature in the proposed system.

CHAPTER II
TECHNICAL BACKGROUND

Conceptual Framework

The "Smart Campus Security: Facial Recognition-Based Entry and Attendance Monitoring System" sought to improve security and automate attendance in educational settings. The concept of the system presented the structural components, such as inputs, processing, and expected outputs, to develop and implement the project.

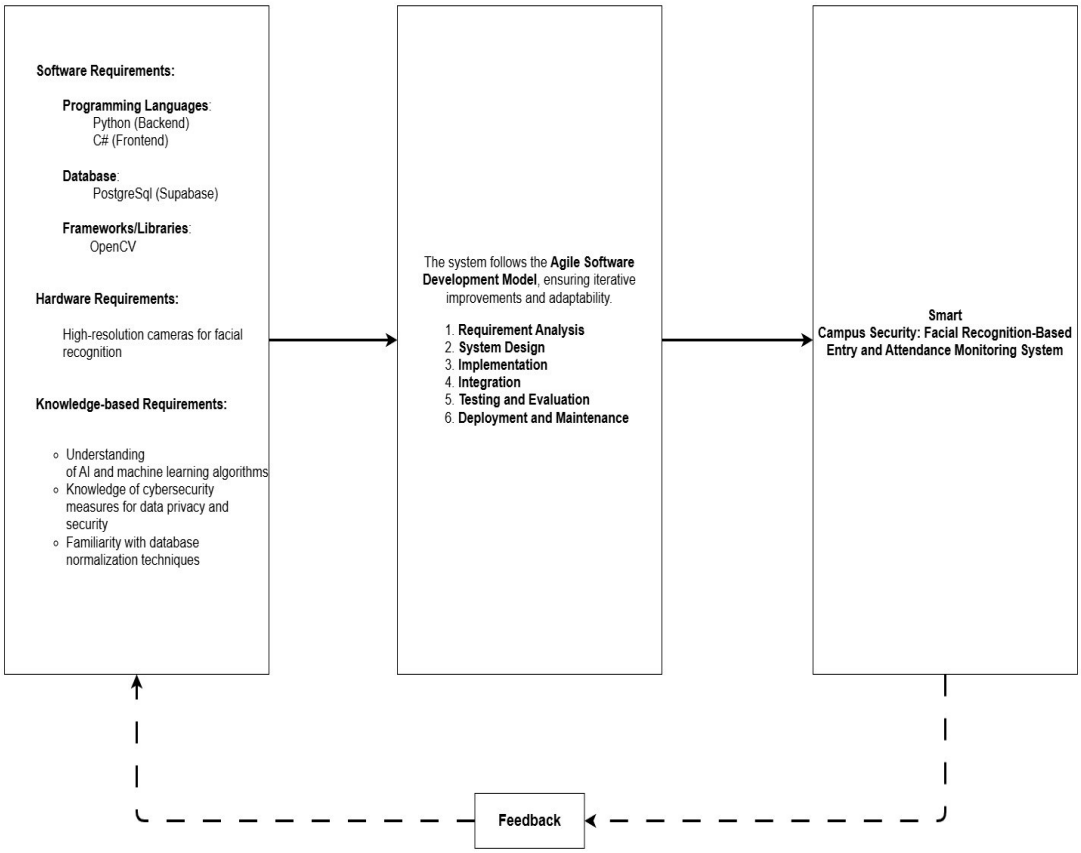


Figure 1. Conceptual Framework

The system followed an Agile Software Development Model to ensure flexibility and collaboration during on-going development. The process began with Requirement Analysis to identify system needs, possible security concerns, and compliance needs. Then the team will produced a System Design, creating the system architecture, UI/UX design and database schema to prepare for development. The next phase was Implementation, which was the coding of backend, frontend, and AI-based facial recognition sysmodule. Once the different components of the system had been

developed, they were connected through Integration. Connection of components included attendance recording, notifications, and security log systems. Next, the system went through Testing and Evaluation, this phase will involved alpha and beta testing. Once the system had been successfully tested, it will undertake Deployment and Maintenance phase. This was the introduction to an educational environment and deployment for ongoing development to meet the needs of users and schools. This development process created a project solution that was systems efficient, secure, and user-centered.

System Architecture

The system architecture defined the overall structure of the system, describing the interaction between components. This included the user interface, processing units, database, and security mechanisms.

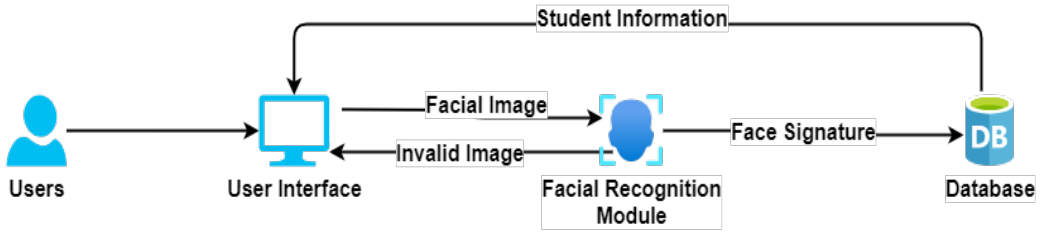


Figure 2. System Architecture

The user interacted with the system through the desktop application, which processed facial features via the Facial Recognition Module. This module scaned the user's face signature against the database and triggered an alert or notification if the user was not registered. Upon successful verification—particularly for students—the system recorded their attendance and displayed their details on the user interface.

Context Diagram

A context diagram represents the interaction between the system and external entities such as students, faculty, administrators, and security personnel.

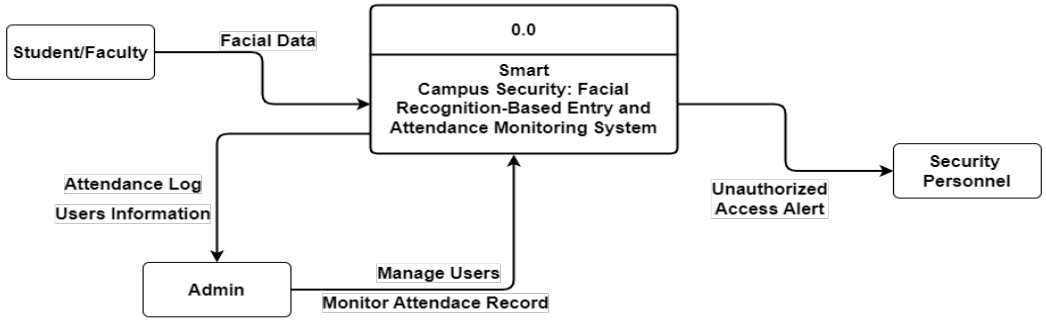


Figure 3. Context Diagram

The system was composed of a number of key components that worked together to provide efficient attendance tracking and management. The first step began with the students and faculty presenting their facial data to serve as the system of authentication and verification of attendance. The admin users managed the system by managing users access and reviewing the attendance logs. The security users received alerts in real time when incidents of unauthenticated access occured, while also ensuring security of the system. The cloud-based data storage securely stored the attendance logs and authenticated logs, served as a single source of truth for data. The facial recognition system was still the technological backbone, secure, seamless, flawless, and accurate user authentication and verification of attendance. Finally, the notification system served to supplement communications to enhance engagement and allow timely notifications and attendance reports to reach their respective stakeholders.

Data Flow Diagram (DFD)

The DFD illustrates how data moved through the system, from input (user authentication) to processing (attendance tracking) and storage (database management).

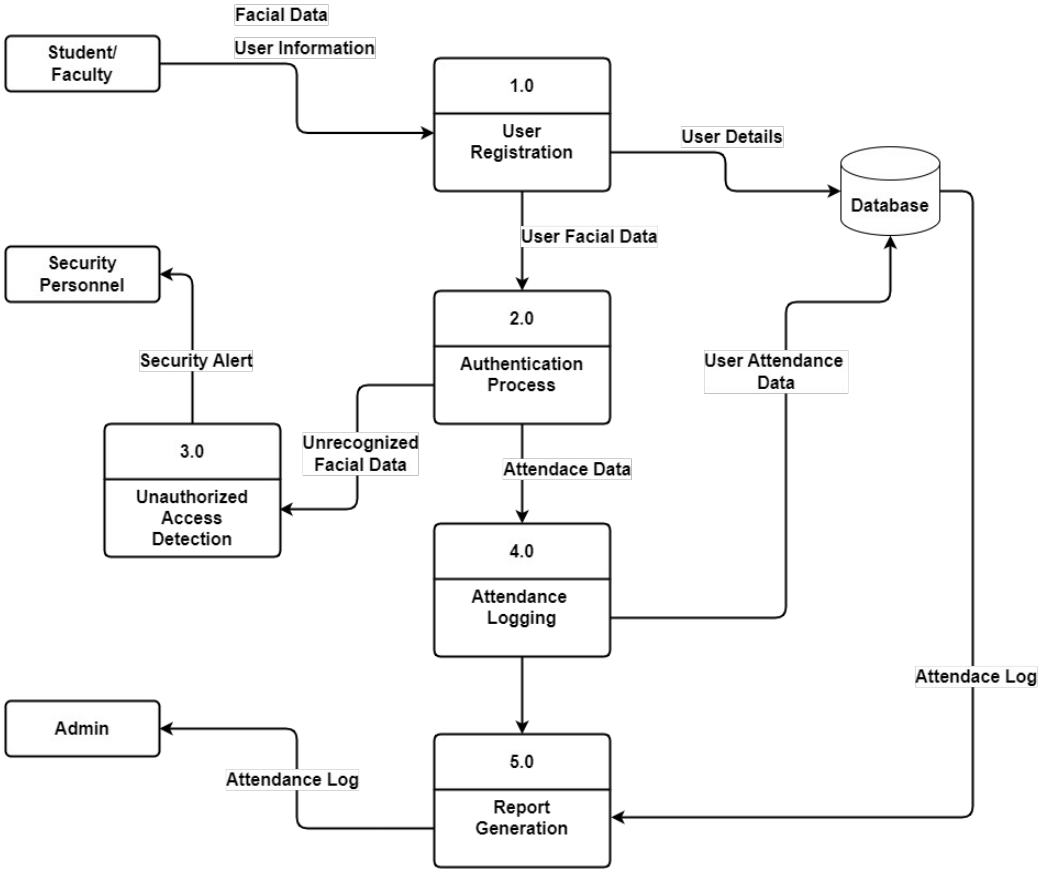


Figure 4. Dataflow Diagram

To guarantee an effective and secure operation, the system's key processes were designed with careful considerations. The process initiated the User Registration Process by securely capturing facial data to use for users during the following process: Authenticating Process. This process was based on real-time facial recognition data, allowing the system to facilitate an accurate verification of each user's identity. The next process was Attendance Logging Process, where attendance records would logged and securely stored to refer to later. In the Report Generation Process, reports were generated of comprehensive attendance records available for authorized faculty and administrators that provide robust information. The system had an Unauthorized Access Detection Process that could alert the security if something did not appear right; such as, if a registered user did not look like the person acknowledged during the user registration process

Entity Relationship Diagram (ERD)

The ERD defines the structure of the system's database, outlining relationships between tables such as users, attendance records, and security logs.

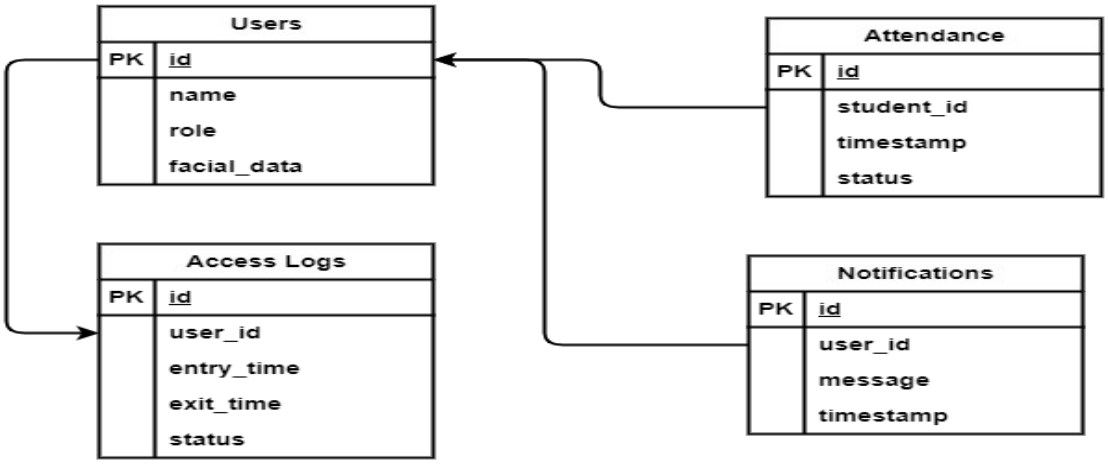


Figure 5. Entity Relationship Diagram

The system's database architecture was designed to maintain efficiency and integrity through well-defined tables and relationships. The Users Table stored details such as student_id, name, role, and facial_data, while the Attendance Table recorded attendance data using fields like attendance_id, student_id, timestamp, and status. The Access Logs Table captured system entry and exit details, including log_id, user_id, entry_time, exit_time, and status. Notifications were managed in the Notifications Table, which logs notification_id, user_id, message, and timestamp. Lastly, the Admin Table held administrative data, including admin_id, name, and privileges.

The relationships among these tables ensured seamless functionality. A one-to-many relationship linked a user to multiple attendance records, reflecting their recurring participation. A one-to-one relationship guarantees that each user was assigned a unique facial recognition profile for authentication. Additionally, a many-to-many relationship connected multiple users with various access events through security logs, ensuring comprehensive tracking and

management of user activity. This structured design supported secure, accurate, and efficient attendance tracking.

Database Schema

The database schema provided a structured representation of the tables, fields, and constraints used in the system.

User Table

The Users Table was the foundation of the database, housing essential details for students and other system users, including facial recognition data for authentication purposes.

Table 1. User Table Schema

Field	Type	Constraints	Example
student_id	SERIAL	PK	1
name	TEXT	NOT NULL	John Doe
role	TEXT	CHECK(role IN ('student', 'admin'))	student
facial_data	BYTEA	NOT NULL	(binary data)

This table securely stores unique user IDs, roles (e.g., student or admin), and essential details like names and facial recognition data, which were key for identity authentication. Each user was linked to their specific facial profile to ensure precise and reliable identification. Additionally, the table supported tracking multiple attendance records for each user, enabling comprehensive monitoring of participation. It played a pivotal role in connecting various components of the system, ensuring streamlined operations and efficient data management.

Attendance Tables

The Attendance Table is designed to log and manage student attendance records accurately. It includes fields for unique attendance IDs, linked student IDs, timestamps for when attendance was recorded, and the status of attendance (e.g., Present, Absent, Late). The table enforces data integrity using a primary key for

attendance_id and a foreign key that links student_id to the Users Table.

Table 2. Attendance Table Schema

Field	Type	Constraints	Example
attendance_id	SERIAL	PK	101
student_id	INT	FK → Users(student_id)	1
timestamp	TIMESTAMP	NOT NULL	2025-03-23 08:00
status	TEXT	CHECK(status IN ('Present', 'Absent', 'Late'))	Present

This table supports a one-to-many relationship, meaning each student can have multiple attendance records over time. For instance, a single student may have records for different days or sessions, ensuring comprehensive tracking of their attendance history. This structure was essential for generating accurate reports and maintaining reliable attendance logs.

Access Logs Table

The Access Logs Table was crucial for tracking user access events, including their entry and exit timestamps. Each log entry was uniquely identified by a log_id, and the table linked each access event to a specific user through the user_id field, which referenced the Users Table. The entry_time field recorded when the user accessed the system, while the exit_time (nullable) captures when they left. The status field indicates the type of access event, such as "Entered" or "Exited," and enforces validity through constraints.

Table 3. Access Table Schema

Field	Type	Constraints	Example
log_id	SERIAL	PK	301
user_id	INT	FK → Users(student_id)	1
entry_time	TIMESTAMP	NOT NULL	2025-03-23 08:05
exit_time	TIMESTAMP	NULLABLE	2025-03-23 15:00
status	TEXT	CHECK(status IN ('Entered', 'Exited'))	Entered

This table supports a many-to-many relationship, as multiple users could be associated with various access events, enabling comprehensive tracking of system activity. The structure ensured reliable logging, allowing for efficient monitoring of user interactions and providing a foundation for generating security and attendance reports.

Notification Table

The Notifications Table was responsible for managing system-generated messages sent to users, ensuring clear communication about events like attendance updates or system alerts. Each notification was uniquely identified by a notification_id, while the user_id field linked the notification to a specific user via a foreign key relationship with the Users Table. The message field stored the content of the notification, and the timestamp field, which was default to the current time (NOW()), recorded when the notification was created.

Table 4. Notification Table Schema

Field	Type	Constraints	Example
notification_id	SERIAL	PK	501
user_id	INT	FK → Users(student_id)	1
message	TEXT	NOT NULL	"Your attendance is recorded."
timestamp	TIMESTAMP	DEFAULT NOW()	2025-03-23 08:10

This table supports a one-to-many relationship, allowing a single user to receive multiple notifications over time. It was essential for keeping users informed and maintained an efficient flow of communication within the system. The Notifications Table integrated seamlessly into the overall database structure, enhancing user engagement and awareness.

Gantt Chart

The Gantt chart presented below outlines the timeline and schedule for the development of the system. This chart provided a visual representation of the various tasks and their durations, as well as the dependencies and milestones throughout the project lifecycle.

Capstone Project Activities						
Legend	Project Start			Project End		
	To Do:		Ongoing:		Done:	

Table 5. Activities for the Month of February

February 2025					
Tasks	**Week 1**				
	M	**T**	**W**	**Th**	**F**
Groupings					
	Week 2				
Brainstorming					
Title Submission					
Title Approval					
Researching for References					
	Week 3				
Starting to Compose Chapter 1					
	Week 4				
Writing and Finalizing Chapter 1 Draft					

Table 6. Activities for the Month of March

March 2025					
Tasks	**Week 1**				
	M	**T**	**W**	**Th**	**F**

Continuing Writing Chapter 1 Draft					
	Week 2				
Finalizing Chapter 1					
Target Submission					
	Week 3				
Adviser Check and Revision					
	Week 4				
Writing Chapter 2					

Table 7. Activities for the Month of April

April 2025					
Tasks	**Week 1**				
	M	**T**	**W**	**Th**	**F**
Continuation of Writing Chapter 2					
	Week 2				
Writing Chapter 3					
	Week 3				
Continuation of Writing Chapter 3					
	Week 4				
Making User Interface					

Table 8. Activities for the Month of May

May 2025					
Tasks	**Week 1**				
	M	**T**	**W**	**Th**	**F**
Making User Interface					
	Week 2				
Finalizing the Papers					

	Week 3				
Defense					

Table 9. Activities for the Month of June

June 2025					
Tasks	**Week 1**				
	M	**T**	**W**	**Th**	**F**
Revision of Chap 1-3					

Table 10. Activities for the Month of July

July 2025					
Tasks	**Week 1**				
	M	**T**	**W**	**Th**	**F**
System Development					
	Week 2				
	Week 3				
	Week 4				

Table 11. Activities for the Month of August

August 2025					
Tasks	**Week 1**				
	M	**T**	**W**	**Th**	**F**
System Development					
	Week 2				
	Week 3				
Start Writing Chapter 4-5					
	Week 4				

System Development					

Table 12. Activities for the Month of September

September 2025					
Tasks	**Week 1**				
	M	**T**	**W**	**Th**	**F**
System Development					
	Week 2				
	Week 3				
Continuation of Chapter 4-5					
	Week 4				
Conduct a Survey					

Table 13. Activities for the Month of October

October 2025					
Tasks	**Week 1**				
	M	**T**	**W**	**Th**	**F**
System Development					
	Week 2				
Conduct a Survey					
	Week 3				
Continuation of Chapter 4-5					
	Week 4				
Finalizing Chapter 4-5					

Table 14. Activities for the Month of November

November 2025					
Tasks	**Week 1**				
	M	**T**	**W**	**Th**	**F**
Checking System and Papers					

	Week 2
Defense	
	Week 3
Revising Paper	
	Week 4

Table 15. Activities for the Month of December

December 2025						
Tasks	**Week 1**					
	M	**T**	**W**	**Th**	**F**	
Revising Paper and Adding Recommendations						
	Week 2					
	Week 3					

Table 16. Activities for the Month of January

January 2026						
Tasks	**Week 1**					
	M	**T**	**W**	**Th**	**F**	
Continuing Revision and Adding Recommendations						
	Week 2					
	Week 3					
	Week 4					

Finalizing the Paper					

Table 17. Activities for the Month of February

February 2026					
Tasks	**Week 1**				
	M	**T**	**W**	**Th**	**F**
Continuing the Finalizing othe Paper					
	Week 2				
Checking the Final Paper					
	Week 3				
	Week 4				
Printing and Hardbinding					

Table 18. Activities for the Month of March

March 2026					
Tasks	**Week 1**				
	M	**T**	**W**	**Th**	**F**
Signing the paper					
	Week 2				
Submission of the Hardbound Thesis					

CHAPTER III
METHODOLOGY

Software Development Methodology

Agile was selected over traditional methodologies like Waterfall due to the dynamic nature of biometric integration. Unlike Waterfall, which required fixed upfront requirements, Agile supported ongoing refinement of modules such as facial recognition algorithms, access control systems, and real-time alerts. This methodology ensured that user feedback from students, faculty, and security personnel could be quickly incorporated into the next development cycle.

Literature in Chapter 2 confirmed that Agile was the most suitable model for AI-enhanced systems (e.g., Lateef & Kamil, 2023; Seah, 2023). Agile's flexibility allowed development teams to fine-tune deep learning models, improve UI/UX design, and conduct continuous security testing across modules like spoofing detection and real-time monitoring. This promoted a high level of system reliability and user satisfaction, both key ISO 25010 attributes. Moreover, Agile fostered early identification of bugs and vulnerabilities, critical for maintaining system integrity in campus environments.

The development process followed a standard Agile lifecycle, represented by the Plan–Design–Develop–Test–Deploy cycle. The

model was sourced and adapted from the Agile Manifesto and was

tailored to support real-time AI model training and feedback integration.

Figure 6. Agile Software Development Methodology

Phases of Agile Development Methodology

The Agile development cycle followed in this study included the following phases:

- **Requirement Analysis:** Identified key security, attendance, and compliance needs from stakeholder interviews and institutional policies.
- **System Design:** Developed system architecture, database schema, and UI wireframes. Integrated cloud-based DBMS with facial recognition logic.
- **Implementation:** Coded front-end (C#) and back-end (Python), connected facial recognition modules to cloud-based databases.
- **Integration:** Linked attendance, facial recognition, and alert systems. Implemented encryption and role-based access control.
- **Testing and Evaluation:** Alpha and Beta Testing. Feedback was incorporated for performance tuning.
- **Deployment and Maintenance:** Pilot tested within a campus environment. Bug reports and feedback informed system updates.

Data Gathering Techniques

To measure the system's alignment with its functional goals and ISO 25010 attributes, structured data collection techniques were implemented. The goal was to evaluate security, usability, reliability, and performance from both user and expert perspectives.

- **Survey (Likert Scale):** Collected quantitative feedback from 42 student/faculty users on usability, performance, and security.
- **Interview:** Conducted with 3 IT professionals to gather technical insights on system reliability, maintainability, and scalability.
- **Document Review:** Analyzed ISO 25010 standards and benchmark studies on AI attendance systems.

Development Requirements

The AI-powered facial recognition system required a combination of computational resources, cloud services, and development tools to handle image processing, identity verification, and secure database interactions.

Table 19. Hardware Requirements

Component	Specification	Purpose
Processor	Intel Core i5 (10th Gen+) / AMD Ryzen 5	Handles real-time facial recognition tasks
RAM	8GB DDR4 (16GB recommended)	Smooth multitasking for AI model inference
Storage	256GB SSD (512GB+ recommended)	Faster data access for model loading
Web Camera	720p HD (1080p preferred) with auto-focus	Clear facial image capture for recognition
GPU (Optional)	NVIDIA GTX 1650 (4GB VRAM) or equivalent	Accelerates deep learning tasks (e.g., TensorFlow)

The necessary hardware components and their specifications for the Smart Campus Security System. It includes recommendations for the processor, RAM, storage, web camera, and an optional GPU, detailing the purpose of each component in

supporting the system's functionality, particularly for facial recognition tasks.

Table 20. Software Requirements

The software tools and frameworks essential for the development and operation of the system. It specifies the programming languages (Python, C#), database systems (PostgreSQL), and version control systems (Git) required for the backend, frontend, and overall project management.

Category	Tools/Frameworks	Purpose
Backend	Python 3.8+ (OpenCV)	Facial recognition, API development
Operating System	Windows 10/11	
Frontend	C# (Winforms)	User interface for attendance dashboard
Database	PostgreSQL 12+ (with pgvector for embeddings)	Secure storage of user data and logs
Version Control	Git, GitHub/GitLab	Collaborative code management

Respondents of the Study

To evaluate the system’s effectiveness and usability, feedback from system users (students/faculty) and technical experts was crucial. Respondent selection was based on Slovin’s formula.

Sampling Size Calculation

Using Slovin’s Formula:

$$n=\frac{N}{1+N(e^2)}$$

Where:

$N=50(total\,accesible\,population)$

$e=0.05(margin\,of\,error\,for\,better\,precision)$

$$n=\frac{50}{1+50(0.05)^2}=\frac{50}{1+50(0.0025)}=\frac{50}{1.125}\approx 44.44$$

Final Sample Size:

To ensure whole respondents, the sample size was **rounded up** to the nearest whole number: **45 respondents**

Table 21. Stratified Respondent Categories:

Group	Sample Size(n)
IT Experts	**3**
End Users	**42**
Total	**45**

Sampling Technique

The study adopted stratified sampling to ensure balanced representation of both user groups — technical evaluators and general users.

This method ensured comprehensive feedback from both end-user experience (ease of use, responsiveness) and technical review (data privacy, anti-spoofing reliability). Stratified sampling minimized selection bias and enhanced evaluation reliability.

Statistical Treatment

To analyze responses to the Likert-scale questionnaire, the Weighted Mean technique was used. This allowed the researchers to quantify feedback across ISO 25010 attributes and derive quality assessments.

Table 22. Likert Scale

Scale	Description	Weight
5	Strongly Agree (Excellent)	5
4	Agree (Good)	4
3	Neutral (Moderate)	3
2	Disagree (Poor)	2
1	Strongly Disagree (Very Poor)	1

Weighted Mean Formula The weighted mean was calculated to determine the average rating for each statement. The formula used is:

$$Weighted\,Mean = \frac{\sum (f x w)}{N}$$

Where:

$f = frequency\,of\,each\,response$

$w = corresponding\,weight (1¿5)$

$N = total\,number\,of\,respondents$

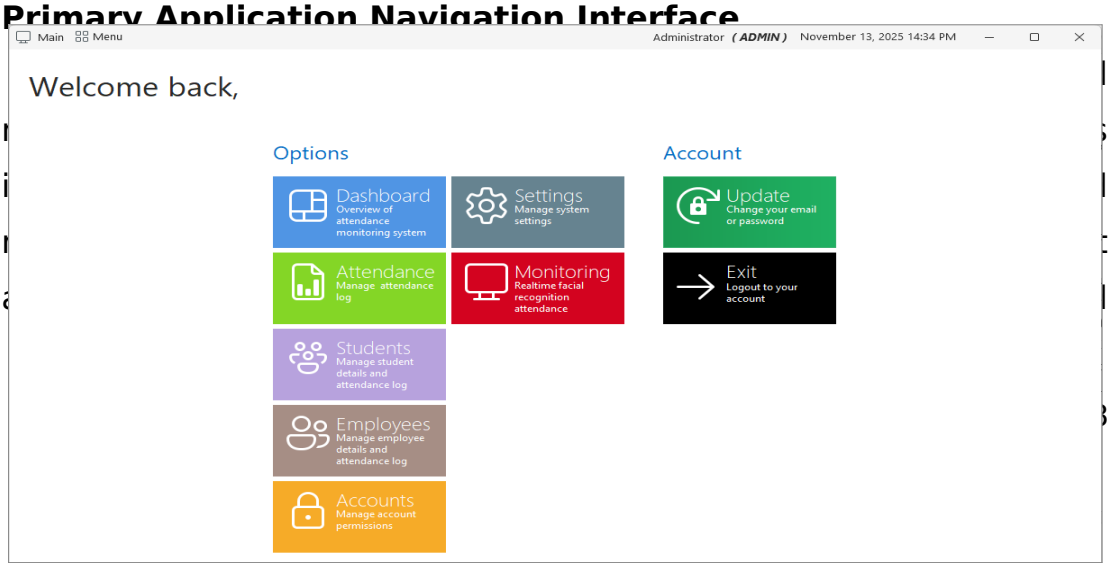
Table 23. Interpretation

Weighted Mean	Description
4.21 – 5.00	Excellent
3.41 – 4.20	Good
2.61 – 3.40	Moderate
1.81 – 2.60	Poor
1.00 – 1.80	Very Poor

CHAPTER IV
RESULTS AND DISCUSSION

This chapter presented the findings from the development and evaluation of the Smart Campus Security System, addressing the study's core objectives. It detailed the realized system through images of its core functionalities, demonstrating the operational facial recognition-based attendance and access control. The results of alpha and beta testing were reported, followed by an analysis of user acceptance surveys. The chapter culminated in a critical evaluation of the system's quality attributes, including performance, security, and usability against the ISO/IEC 25010 standard, providing a validated assessment of the system's efficacy and compliance.

Primary Application Navigation Interface



for real-time facial recognition, and essential user utilities such as the password update function and the application exit command.

Figure 7. Main Application Menu

System Settings

The System Settings module provided a centralized interface for the comprehensive administration and configuration of all institutional and operational parameters. This included the management of fundamental academic and organizational structures—such as Year Level, Block, Course, and Employee Position—as well as the critical configuration of user security through Account Roles and Permissions. Furthermore, this module served as the access point for the specialized Facial Recognition Configuration, consolidating all essential controled into a single

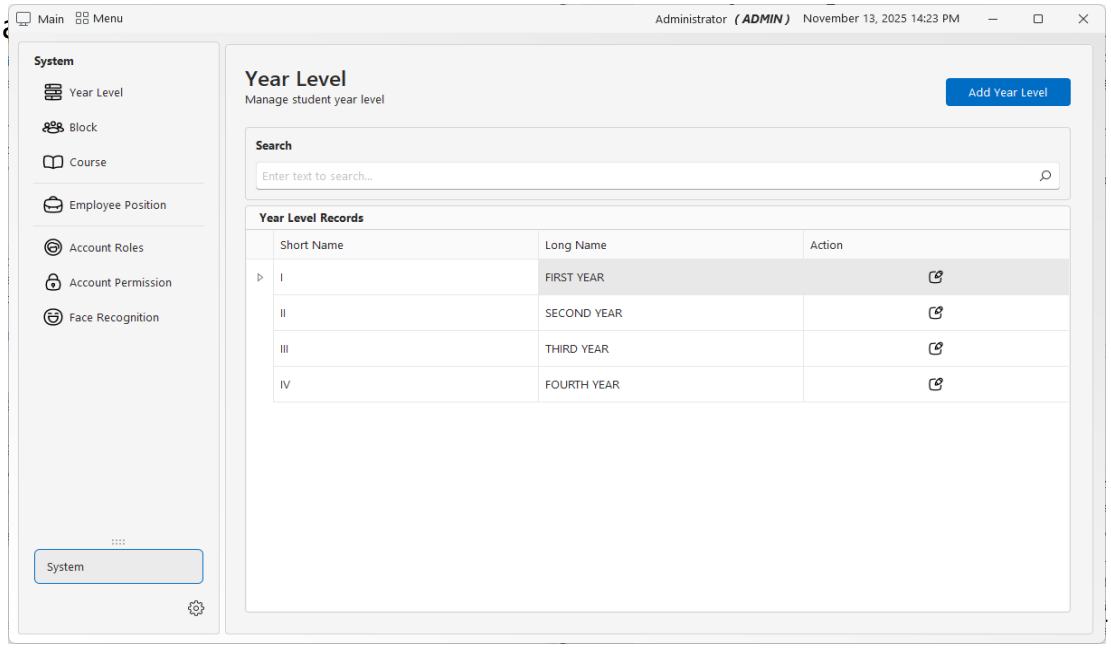
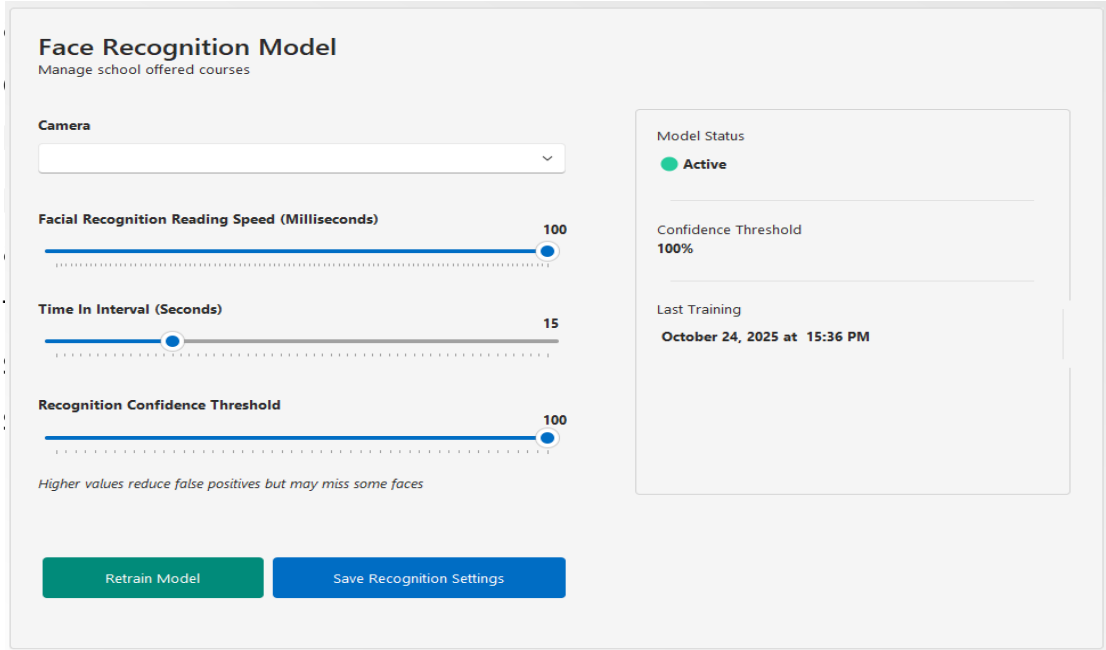
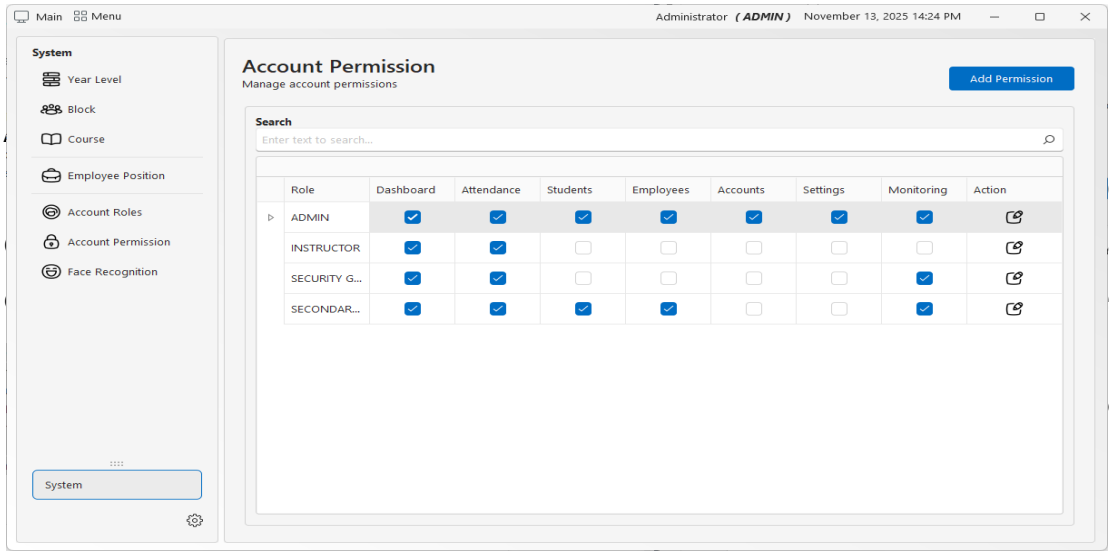


Figure 8. System Settings

System Facial Recognition Configuration

The Facial Recognition Configuration module served as the administrative core of the system. This interface allowed





system's security architecture, ensuring that users could only access data and functions pertinent to their roles, thereby mitigating the risk of unauthorized data exposure and security breaches.

Figure 10. Accou

nt Permission

Student Registration Module

The Student Registration Module served as the initial enrollment point within the system, where students input their personal and academic details, including name, student ID, course, and year level. Integrated directly into this process was a facial data capture interface, which captured their images. This collected visual data were subsequently processed and used to train the facial recognition model, thereby registering the individual as a recognized entity for subsequent authentication and attendance

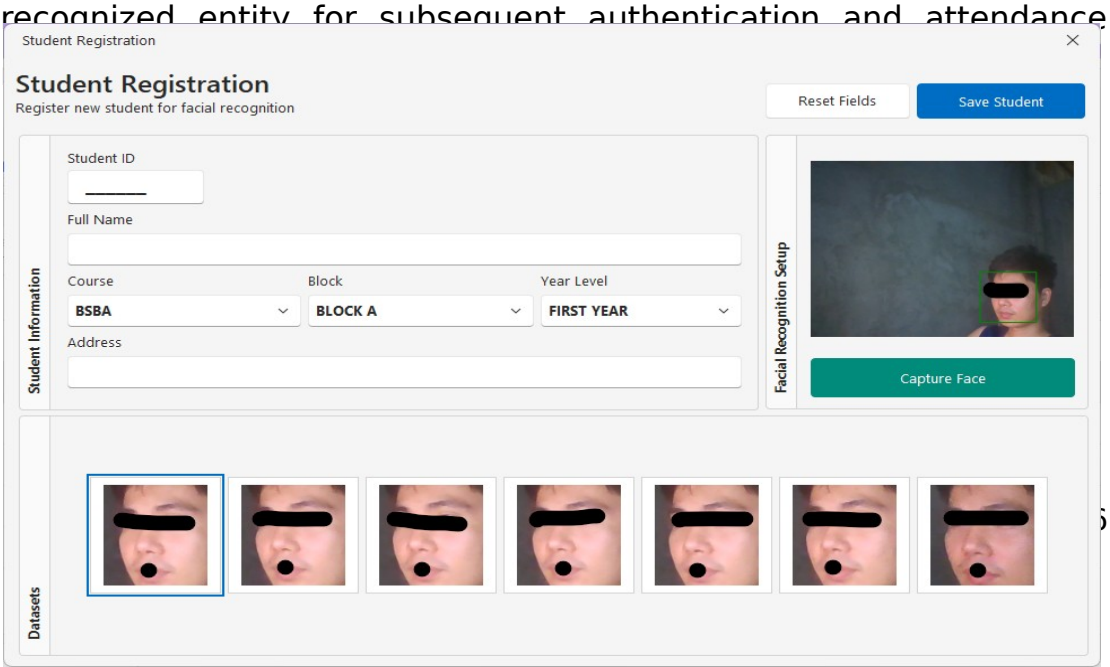


Figure 11. Student Registration

Monitoring Module

The Monitoring Module constituted the operational core of the Smart Campus Security System, providing a real-time interface for automated attendance logging and access control. This module was responsible for processing the live video stream from designated cameras, upon which it executed a continuous cycle of facial detection and recognition. Upon a successful identification, the system automatically logged the individual's entry or exit, recorded

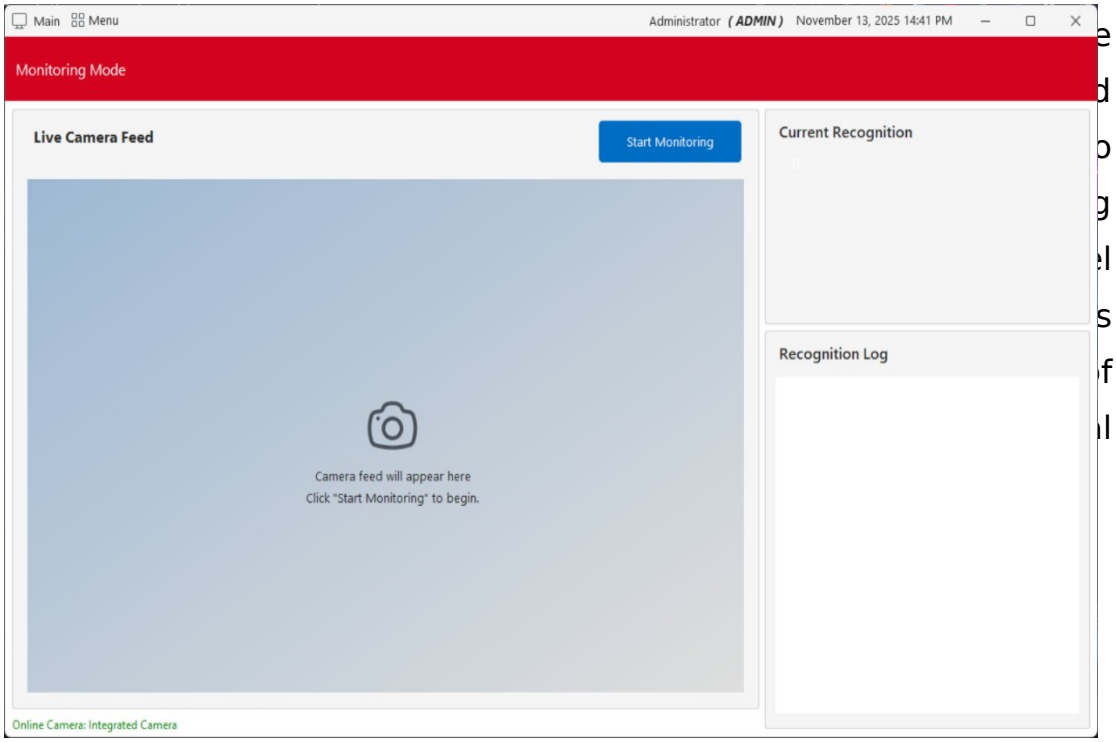


Figure 12. Monitoring Module

System Testing: Alpha and Beta Phases

This section outlined the empirical evaluation of the Smart Campus Security System through structured Alpha and Beta testing phases. The Alpha testing was conducted internally in a controlled laboratory setting to identify and rectify critical functional, performance, and security flaws. Subsequently, Beta testing was deployed in a live campus environment with a select group of end-users. This phase aimed to validate the system's reliability, usability, and robustness under real-world conditions, while also gathering qualitative feedback on the user experience. The results from these sequential testing phases provided critical insights into the system's operational readiness and practical efficacy prior to full-scale deployment.

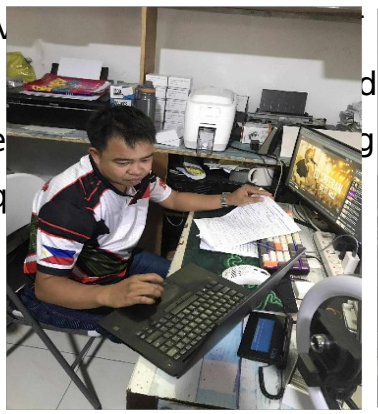
Alpha Testing

The accompanying image documented the initial Alpha testing phase, conducted internally by the system developers. This critical first stage of evaluation was performed in a controlled laboratory environment, focusing on the identification and resolution of functional, performance, and logical errors. The objective was to rigorously stress-test all core components—including facial recognition accuracy, database integration, and user interface responsiveness—before exposing the system to external users. This process ensured the stabilization of the core software and the remediation of critical flaws, establishing a foundational level of

Figure 13. Alpha Testing

Beta Testing

The Beta testing phase involved deploying the functional system to a representative group of external users, including



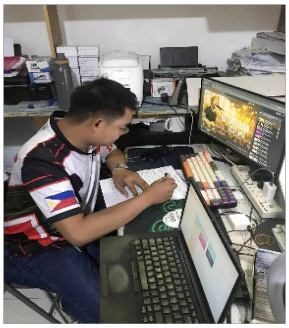


Figure 14. Beta Testing

The figure captures this user-acceptance testing, showing the practical demonstration and data collection process with the various stakeholder groups. The results from this phase provide critical, validated insights into user satisfaction and the system's compliance with established software quality standards.

Evaluation using ISO 25010

This section presents a formal evaluation of the Smart Campus Security System against the ISO/IEC 25010 software quality model. The system's critical attributes including its functional suitability, performance efficiency, compatibility, usability, reliability, security, and portability are systematically assessed. This evaluation synthesized quantitative data from system tests and qualitative feedback from beta testing questionnaires to provide a standardized, holistic measure of the software's quality. The analysis verified the system's compliance with international standards and identified key strengths and areas for future improvement, offered a validated conclusion regarding its overall efficacy and robustness.

Table 24. Functional Suitability

A. Functional Suitability		
Criteria	**Mean**	**Description**
1. The system accurately verifies student and faculty identities.	3.93	Good

2. The system records attendance without requiring manual intervention.	3.69	Good
TOTAL	**3.81**	**Good**

The Functional Suitability evaluation yielded an overall mean of 3.81, categorizing the system's performance as "Good." This result indicated robust fulfillment of core objectives, with the near-excellent score for identity verification (3.93) confirming the efficacy of the facial recognition engine. The slightly lower, yet still positive, score for automated attendance logging (3.69) suggested the process was reliable but identified a specific area for potential refinement to enhance perceived seamlessness. The data conclusively validated that the system met its fundamental functional requirements as perceived by end-users.

Table 25. Performance Efficiency

B. Performance Efficiency		
Criteria	**Mean**	**Description**
1. The facial recognition feature operates in real time.	3.84	Good
2. The system provides satisfactory response time.	3.76	Good
TOTAL	**3.80**	**Good**

The Performance Efficiency results indicated a consistently satisfactory system response. The overall mean of **3.80** confirmed proficient operational speed. The marginally higher score for real-time facial recognition **(3.84)** versus general system response **(3.76)** suggested the core biometric algorithm was highly optimized, while ancillary processes presented a minor latency that could be targeted for further performance tuning.

Table 26. Portability

C. Portability

Criteria	Mean	Description
1. The system operates smoothly across different windows devices.	3.44	Good
2. Cloud integration remains stable across platforms.	3.73	Good
TOTAL	**3.59**	**Good**

The Portability evaluation, which assesses system adaptability across different environments, yields a positive overall result. The total mean of **3.59** confirms good cross-platform functionality. The data indicated a key strength in cloud integration stability, which received a mean score of **3.73**. However, a notable variance was observed in interoperability, as performance across different Windows devices received a lower mean of **3.44**. This suggested potential driver dependencies or hardware-specific performance inconsistencies that slightly constrain seamless deployment.

Table 27. Compatibility

D. Compatibility		
Criteria	**Mean**	**Description**
1. The system integrates well with existing school databases and systems.	3.49	Good
TOTAL	**3.49**	**Good**
E. Usability		
Criteria	**Mean**	**Description**
1. The user interface is intuitive and easy to navigate.	4.82	Excellent
2. Instructions and labels are clear and easy to understand.	4.00	Good

TOTAL	**4.41**	**Excellent**

The evaluation of the Compatibility characteristic, which measured the system's ability to exchange information and co-operate with other systems, yields a positive result. The mean score of **3.49** fells within the "Good" descriptor. This indicated that the system was perceived to integrate satisfactorily with existing school databases and IT infrastructure. While the score confirmed functional interoperability, its position in the lower range of the "Good" band suggested there was measurable room for improvement in achieving more seamless and robust data exchange.

Table 28. Usability

The Usability evaluation yields an outstanding result, with an overall mean of **4.41**, which fell squarely within the "Excellent" classification. This indicated a highly positive user perception of the system's interface and learnability. A significant strength was the interface's intuitiveness, which received a near-perfect score of **4.82**. While the clarity of instructions and labels also scored positively at **4.00**, the notable variance between the two criteria suggested that refining documentation and on-screen guidance presented a clear opportunity to elevate the user experience from excellent to exceptional.

Table 29. Reliability

F. Reliability		
Criteria	**Mean**	**Description**
1. The system performs consistently during repeated use.	3.76	Good
2. Attendance logs are generated accurately and reliably.	4.00	Good
TOTAL	**3.88**	**Good**

The Reliability assessment, which measures the system's ability to maintain its level of performance under normal use, yielded a strong positive result. The overall mean of **3.88** was rated as "Good." The data indicated a high degree of confidence in the accuracy of the system's core output, as evidenced by the "Excellent" rating for log accuracy (**Mean = 4.00**). The slightly lower, though still robust, score for operational consistency (**Mean = 3.76**) suggested that while the system's results were trusted, users may perceived minor fluctuations in performance during extended use, identifying an area for further stabilization.

Table 30. Security

G. Security		
Criteria	**Mean**	**Description**
1. User data is securely stored and protected from unauthorized access.	3.69	Good
2. The system effectively detects spoofing or fake identities.	3.91	Good
TOTAL	**3.80**	**Good**

The Security evaluation indicated a robust and trustworthy system, with an overall mean of **3.80** falling within the "Good" classification. The system's ability to detect spoofing was perceived as a particular strength, scoring **3.91** and approaching the "Excellent" threshold. The score for data protection, while also strong at **3.69**, suggested that users perceived this aspect as slightly less assured. This divergence pointed to a key finding: the biometric security was highly effective, but perceptions of backend data security could be enhanced through reinforced transparency or additional safeguards.

Table 31. Maintainability

H. Maintainability		
Criteria	**Mean**	**Description**
1. The code is modular and allows for easy updates.	3.93	Good
2. System errors can be resolved efficiently.	3.67	Good
TOTAL	**3.80**	**Good**

The Maintainability assessment yielded a positive result, with an overall mean of **3.80** indicating a "Good" capacity for future updates and error correction. The data revealed a key strength in the system's foundational architecture, as evidenced by the high score for modular code (**Mean = 3.93**). This suggested the codebase was well-structured for incorporating new features. The lower score for efficient error resolution (**Mean = 3.67**), while still positive, indicated that the processes for diagnosing and fixing system errors were perceived as less streamlined than the code structure itself, presenting a clear area for procedural improvement.

CHAPTER V
SUMMARY, CONCLUSION AND RECOMMENDATIONS

This chapter served as the culmination of the research and development process, providing a comprehensive synthesis of the project's outcomes. It systematically summarized the key findings

derived from the system's development and empirical evaluation, formulated definitive conclusions grounded in the ISO/IEC 25010 quality assessment, and proposed targeted recommendations for both the immediate refinement and long-term evolution of the system.

Summary

The present study entailed the end-to-end design, development, and rigorous evaluation of a Smart Campus Security System architected around facial recognition technology. The implemented system comprised a suite of integrated modules—including Student Registration, Facial Recognition Configuration, Real-time Monitoring, and Administrative Management—that collectively fulfilled the core objectives of automating attendance logging and enforcing access control. The developmental lifecycle was validated through a structured testing regimen. Initial Alpha testing confirmed the stability of core functionalities in a controlled environment. Subsequent Beta testing, conducted in an operational campus setting with stakeholders, provided critical data on real-world usage. The principal findings were derived from a quantitative evaluation based on the ISO/IEC 25010 quality model, which yielded the following key results:

- **High User-Centric Performance**: The system achieved an "Excellent" rating in Usability (Weighted Mean = 4.41), with the user interface (UI) intuitiveness scoring a near-perfect 4.82. This indicated superior user experience and learnability.
- **Robust Core Functionality**: The system demonstrated "Good" to "Excellent" performance in its primary functions. Reliability (WM=3.88) and Functional Suitability (WM=3.81) were confirmed, with attendance log accuracy scoring 4.00 ("Excellent").
- **Adequate Operational and Structural Qualities:** The system's PerformanceEfficiency (WM=3.80), Security (WM=3.

80), and Maintainability (WM=3.80) were all rated as "Good," indicating a solid, well-rounded foundation.

- **Identified Areas for Optimization**: The evaluation pinpointed specific constraints. Compatibility (WM=3.49) and specific aspects of Portability (WM=3.59) scored in the lower "Good" range, suggesting dependencies on specific hardware or software environments that hindered seamless interoperability.

Conclusion

Based on the empirical evidence gathered through the ISO/IEC 25010 evaluation, it was concluded that the developed Smart Campus Security System was a technically sound, reliable, and user-acceptable solution for modernizing campus security and attendance monitoring protocols. The system successfully met its specified functional requirements, delivering accurate, automated, and contactless authentication. The high scores in Usability and Functional Suitability are indicative of a successful design that aligns with end-user needs and expectations. Furthermore, the "Good" ratings across Performance Efficiency, Security, and Maintainability affirm that the system was not only operationally competent but also possesses a codebase amenable to future modification and scaling. While the evaluation confirmed overall efficacy, it also objectively identified non-critical deficiencies in cross-platform compatibility and error resolution workflows, which do not preclude deployment but represent clear targets for iterative improvement.

Recommendations

To advance the system from a functional prototype to a robust, enterprise-grade application, the following recommendations were stratified into immediate technical enhancements and long-term research trajectories.

Recommendations for Immediate System Enhancement

1. **Optimize Asynchronous Processing Pipelines:** To address the marginal latency in general system response (Mean=3.76), a profiling of the application's backend services should be conducted. Optimizing database query execution, implementing caching mechanisms for frequently accessed data (e.g., student profiles), and leveraging asynchronous programming for non-critical tasks (e.g., log writing) are recommended to elevate Performance Efficiency.

2. **Decouple Hardware Dependencies:** To improve Portability scores across Windows devices (Mean=3.44), it is recommended to abstract the camera interaction layer. Implementing a standardized API (e.g., using OpenCV's DirectShow or Media Foundation backends) and creating a hardware compatibility suite for pre-deployment checks would mitigate device-specific inconsistencies.

3. **Implement a Structured Logging and Alerting Framework:** To enhance the efficiency of error resolution (Maintainability, Mean=3.67), the integration of a centralized, structured logging system (e.g., using the ELK Stack - Elasticsearch, Logstash, Kibana) is proposed. This will facilitate real-time monitoring, automated alerting for critical exceptions, and significantly reduce mean time to resolution (MTTR).

Recommendations for Future Research and Development

1. **Integrate Multi-Modal Biometric Authentication:** To fortify security against sophisticated spoofing attacks, future work should investigate the fusion of facial recognition with a secondary biometric modality, such as gait analysis or thermal imaging, to create a robust liveness detection system.

2. **Develop Predictive Analytics Capabilities:** Leveraging the accumulated attendance data, a promising research direction

is the development of a machine learning module for predictive analytics. This could identify patterns of student engagement, predict at-risk students based on attendance anomalies, and provide actionable insights for academic intervention.

3. **Architect a Microservices-Based Deployment:** To address future scalability challenges, a longitudinal study should be initiated to redesign the current monolithic architecture into a containerized microservices model (e.g., using Docker and Kubernetes). This would enhance scalability, facilitate independent deployment of services, and improve overall system resilience.

BIBLIOGRAPHY

References

Tan, S. J. (2018). Facial recognition-based attendance monitoring system for educational institution (Doctoral dissertation, UTAR).

Pramod Kumar, P., Akshay, R., & Sagar, K. (2023, October). Face Recognition-Based Smart Attendance Monitoring System in Classroom. In International Conference on Trends in Sustainable Computing and Machine Intelligence (pp. 407-422). Singapore: Springer Nature Singapore.

GUDIKANDHULA, N. R., Kondapalli, B., Acharyulu, M. L. N., Sameeri, M., Supradeep, D., & Yaswanth, Y. (2023). Enhancing University Security: A Machine Learning and IoT Driven Face Recognition System for Surveillance and Attendance.

El-Mashad, Y., & Ali, H. A. (2024). A new approach for smart attendance system based on improved video facial recognition technology for smart university.

Thaleeparambil, Noeal & Biju, Abiya & Rudra Boppuru, Prathap. (2024). Integrated Automated Attendance System with RFID, Wi-Fi, and Visual Recognition Technology for Enhanced Classroom Security and Precise Monitoring. 10.1109/InC460750.2024.10649192.

Seah, Y. (2023). Smart attendance system with Iot based face recognition using deep learning approach (Doctoral dissertation, UTAR).

Thirukrishna, J. T., Revathi, A. M., Shashank, Y., Pandith, T., & Samarth, N. (2023). Smart Attendance System Using Face Recognition. Asian Journal of Engineering and Applied Technology, 12(2), 34-39.

https://asana.com/id/resources/agile-methodology

Karthick, V., Kumar, P., Divyaprasath, R., & Kesarinath, K. (2024). Facial Recognition Based Attendance Management System. Available at SSRN 4834649.

BOURDIMA, Y. (2024). Enhancing Campus Security: Face Detection and Automatic Recognition System for Guelma University.

Zainuddin, A. A., Nor, R. M., Handayani, D., Tamrin, M. I. M., Subramaniam, K., & Sadikan, S. F. N. (2024). Smart Attendance in Classroom (CObot): IoT and Facial Recognition for Educational and Entrepreneurial Impact. Aptisi Transactions on Technopreneurship (ATT), 6(3), 608-622.

Lateef, A. S., & Kamil, M. Y. (2023). Facial Recognition Technology-Based Attendance Management System Application in Smart Classroom. Iraqi Journal For Computer Science and Mathematics, 4(3), 136-158.

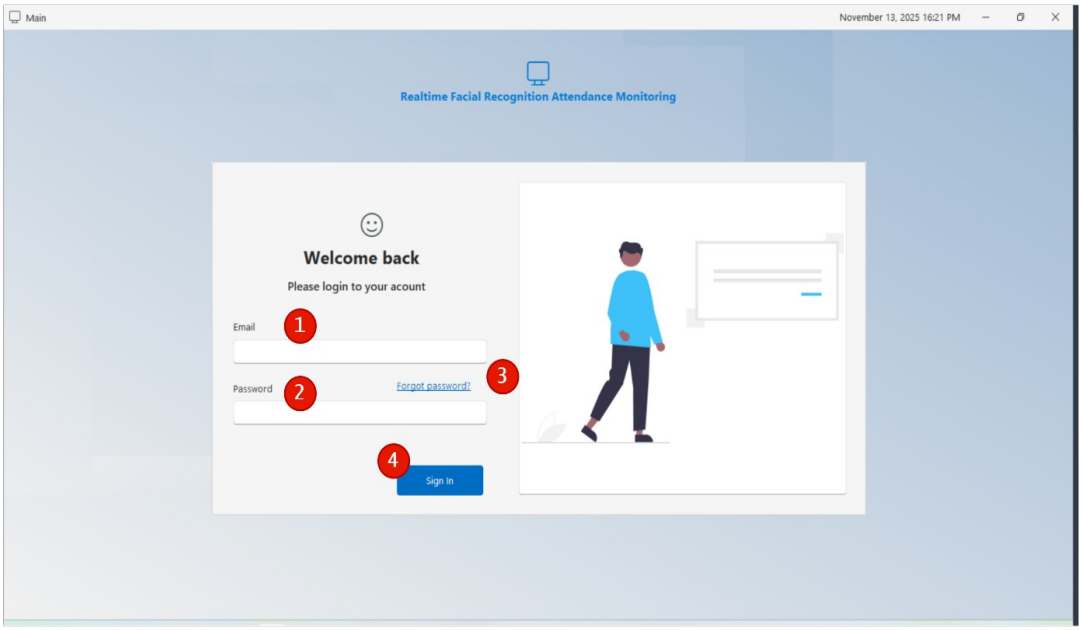
Rajesh, T. R., Kuchipudi, L. N., Dammalapati, A., & Surendran, R. (2024, July). Attendance System Based on Facial Recognition Using OpenCV. In 2024 2nd International Conference on Sustainable Computing and Smart Systems (ICSCSS) (pp. 1529-1535). IEEE.

Patel, S., Kumar, P., Garg, S., & Kumar, R. (2018). Face Recognition based smart attendance system using IOT. International Journal of Computer Sciences and Engineering, 6(5), 871-877.

APPENDIX A: USER'S MANUAL

Welcome to the User Manual for the **Smart Campus Security System**. This guide is designed to assist **Administrators** in effectively managing the system and leveraging its facial recognition technology to automate campus security and attendance monitoring. You will find clear step-by-step instructions on how to configure the facial recognition settings, register students and employees, monitor real-time access logs, and manage system permissions. This manual serves as your comprehensive resource for utilizing the system to enhance campus safety, streamline administrative processes, and maintain accurate, contactless attendance records.

Login Interface

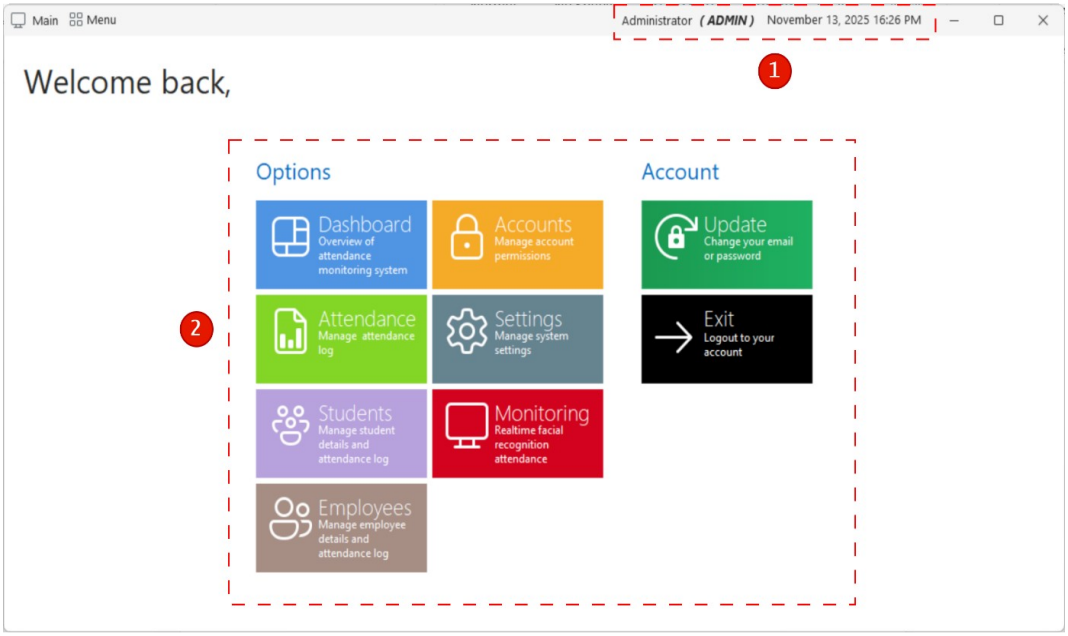


Login Page	
1. Email Input Box	In the Email field, enter your registered email address. This is the email you used during account creation.
2. Password Input Box	In the Password field, type in your secure password. Make sure to enter it correctly, as passwords are case-sensitive.
3. Forgot	If you've forgotten your password, click this link

Password Link	to begin the password recovery process. You will be guided through steps to reset your password via email.
4. Login Button	After entering your email and password, click on the Login button to proceed with authentication.
5. Successful Login	• If your credentials are correct, you will be redirected to the system's main dashboard. • You can now access features and tools based on your assigned user role.

System Main Menu

The Main Menu was the central navigation hub of the system, providing access to all core modules and user functions.

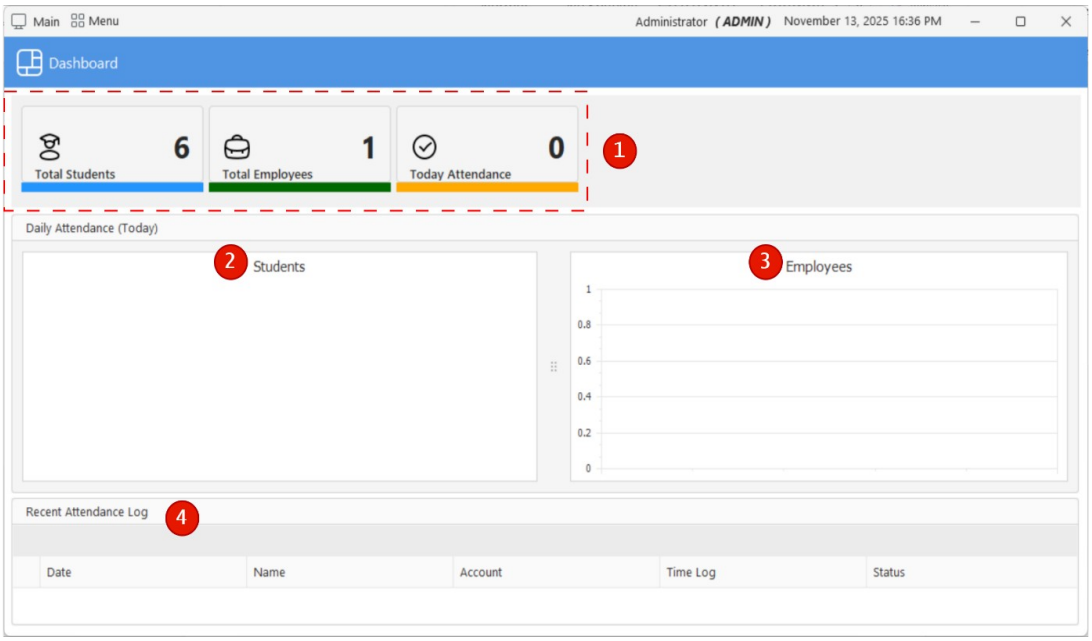


Main Menu	
1. Account Profile	This section allows you to view and manage your personal account information. You can update your contact details and securely change your account password here.
2. System Menu	This is the primary access point for all administrative and operational functions of the security system. Clicking this will reveal a

	comprehensive menu, including: • **Dashboard:** For an overview of system activity. • **Attendance Log:** To review access history. • **Students/Employees:** To manage user registrations. • **Monitoring:** For the live video feed. • **Accounts & Settings:** For system configuration (Admin only).

Dashboard

The Dashboard provides a comprehensive, at-a-glance overview of the system's key metrics and recent activity.

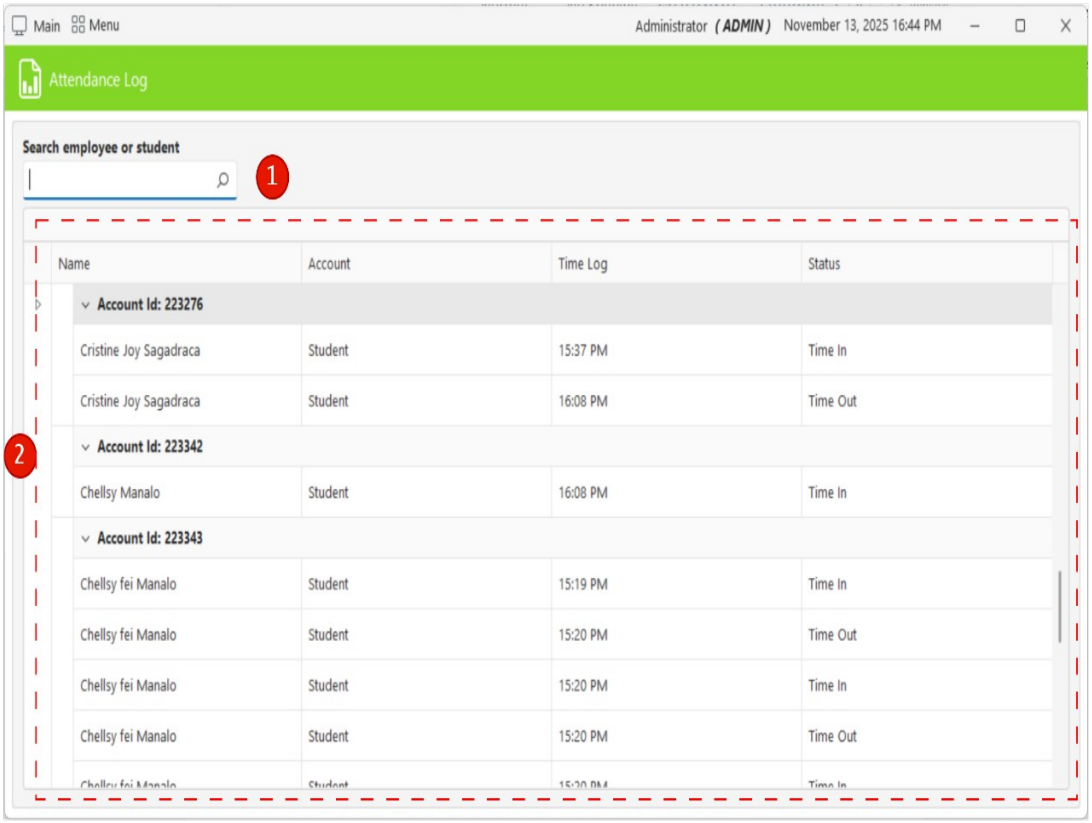


Dashboard	
1. Statistics Panel	This section displays real-time summary statistics, including the total number of registered **Students**, registered **Employees**, and the **Total Attendance** count recorded for the current day.
2. Student	This visual graph presents student attendance

Attendance Graph	metrics, segmented and displayed by individual academic **Courses**. It allows for quick analysis of attendance patterns across different student groups.
3. Employee Attendance Graph	This visual graph presents employee attendance metrics, categorized by **Employee Positions**. It provides insights into staff presence and distribution.
4. Recent Attendance Log	This panel shows a chronological record of the most recent attendance entries, listing each event by the name of the **Student** or **Employee**, along with the corresponding timestamp and status (e.g., Time-In/Time-Out).

Attendance Log

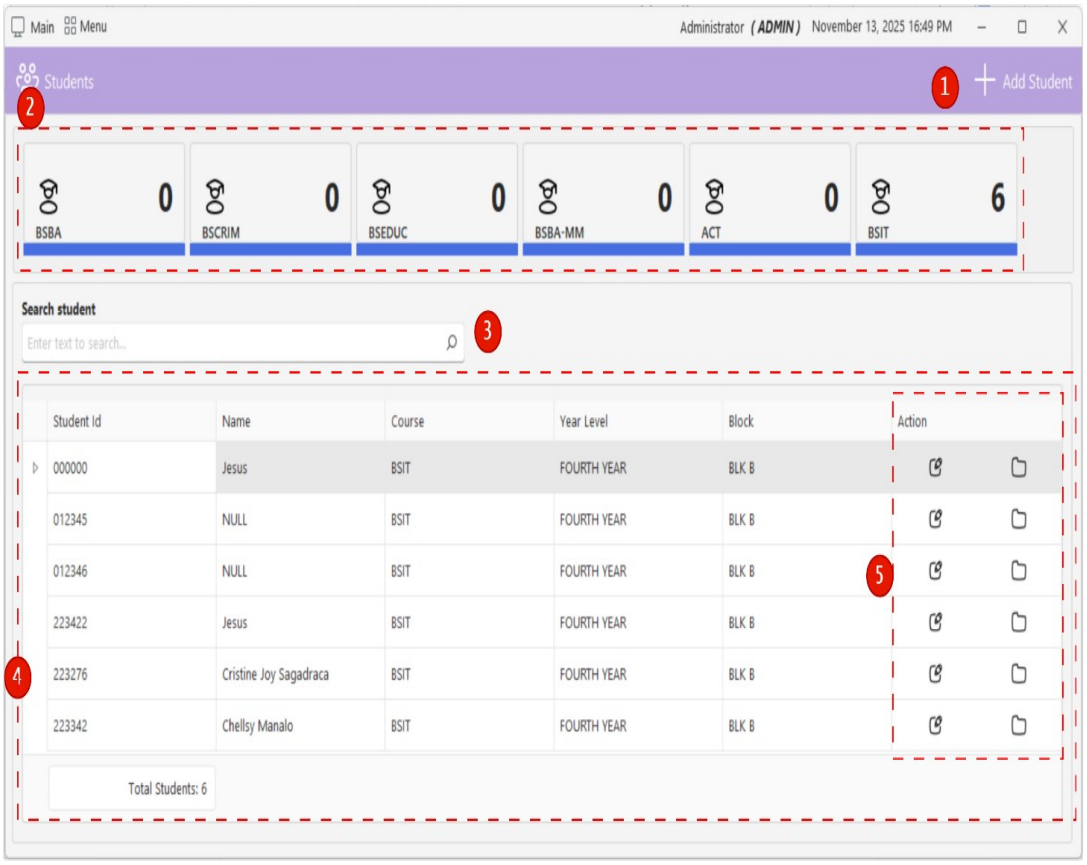
The Attendance Log module provides a comprehensive and searchable record of all access events captured by the system.



Attendance Log	
1. Search Input Box	This field allows you to search and filter the list of attendance records. You can enter a name, student ID, or other identifying information to quickly locate specific entries.
2. Attendance Grid View List	This is the main display area, presenting all attendance records in a structured table. The list typically includes details such as: • **Name** of the student or employee. • **Identification Number** (e.g., Student ID, Employee ID). • **Timestamp** of the entry/exit event. • **Status** (e.g., "Time-In," "Time-Out").

Student Page

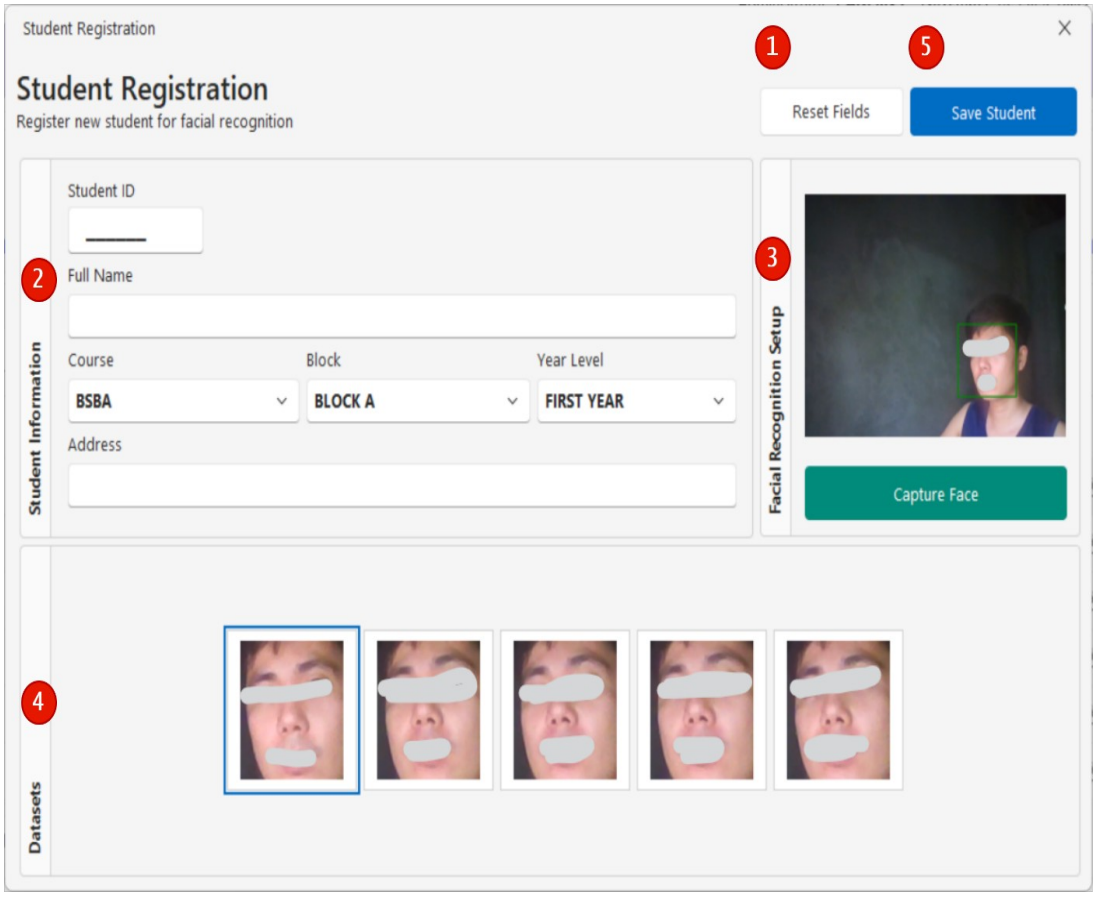
The Student Page is the central module for managing all student-related data and registrations within the system.



Student Page	
1. Add Student Button	This button opens the student registration form, allowing an administrator to enroll a new student by entering their details and capturing their facial data.
2. Course Stats Card	This widget displays a summary statistic, showing the total number of registered students categorized by each academic **Course**.
3. Search Input Box	This field is used to quickly locate a specific student within the list by searching against their name, student ID, or other relevant identifiers.
4. Student Record List	This is the main tabular view that displays a list of all registered students, showing key information such as Name, Student ID, Course, and Year Level.
5. Action Button	These are context-sensitive buttons located in each student record. • The first button allows you to **modify** the student's personal and academic information. • The second button allows you to **view** the detailed attendance history log for that specific student.

Student Registration Form

This form is used to enroll a new student into the Smart Campus Security System, capturing both their personal information and facial data for the recognition model.

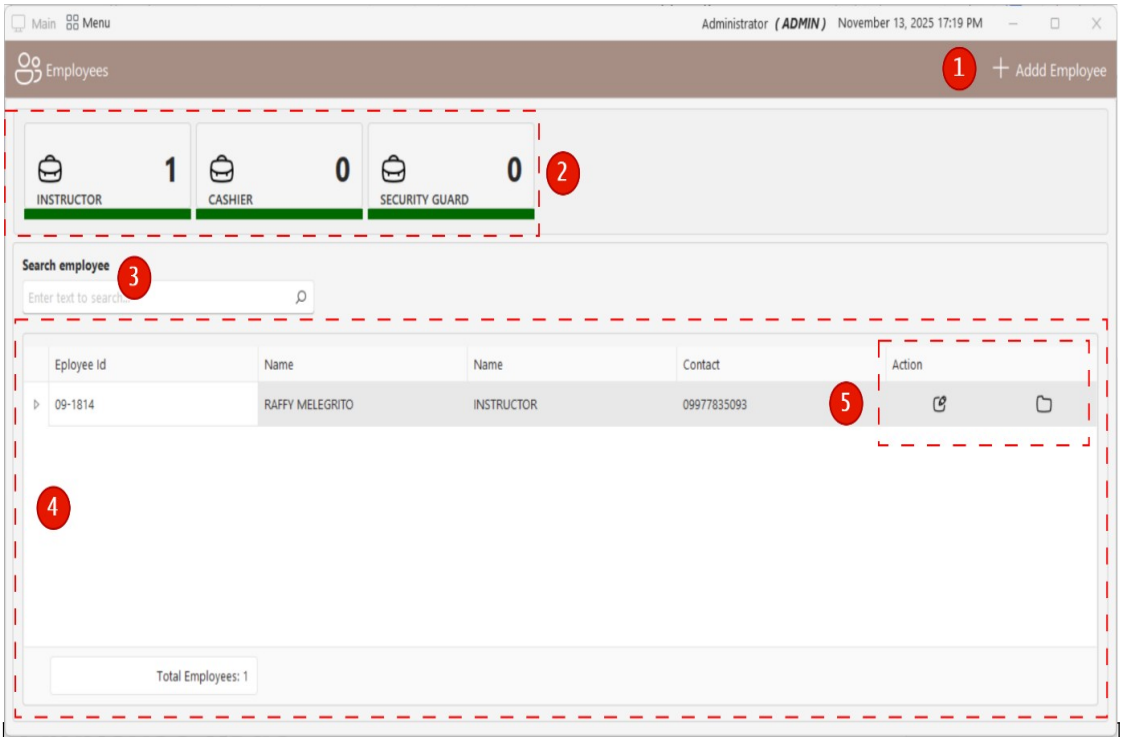


Student Page	
1. Reset Button	Click this button to clear all entered data in the Student Information fields and permanently delete all captured face images from the current session. Use this to start the registration process over.
2. Student Information Panel	This section contains the required fields for entering the student's demographic and academic data. The fields typically include: • **Student ID** • **Full Name** • **Course** • **Block** • **Year Level** • **Address**
3. Live Camera	This panel displays a real-time video feed

Feed	from the selected camera. Below the feed, a **'Capture' button** is used to take snapshots of the student's face from different angles to build a robust dataset.
4. Face Dataset Panel	This area displays a gallery of all the facial images captured for the current student registration. These images are used to train the facial recognition model for that specific individual.
5. Save Button	Click this button to finalize the registration process. It saves all the provided student information and the associated face dataset to the system database, making the student recognizable by the facial recognition system.

Employee Page

The Employee Page is the central module for managing all employee-related data and registrations within the system.



Employee Page

1. Add Employee Button	This button opens the student registration form, allowing an administrator to register a new employee by entering their details and capturing their facial data.
2. Position Stats Card	This widget displays a summary statistic, showing the total number of registered employees categorized by each positions.
3. Search Input Box	This field is used to quickly locate a specific employee within the list by searching against their name, student ID, or other relevant identifiers.
4. Employee Record List	This is the main tabular view that displays a list of all registered employees, showing key information such as Name, Position and Employee ID.
5. Action Button	These are context-sensitive buttons located in each student record. • The first button allows you to **modify** the employee's personal information. • The second button allows you to **view** the detailed attendance history log for that specific employee.

APPENDIX B: SURVEY QUESTIONNAIRE

Direction: Please evaluate the developed system – **Smart Campus Security: Facial Recognition-Based Entry and Attendance Monitoring System** by putting a checkmark [✔] on the column that corresponds to your assessment. Use the scale below:

5 – Strongly Agree	**4 – Agree**	**3 – Neutral**
2 – Disagree	**1 – Strongly Disagree**	

Criteria	**Scale**				
Functional Suitability	**1**	**2**	**3**	**4**	**5**
The system accurately verifies student and faculty identities.					
The system records attendance without requiring manual intervention.					
Performance Efficiency	**1**	**2**	**3**	**4**	**5**
The facial recognition feature operates in real time.					
The system provides satisfactory response time.					
Portability	**1**	**2**	**3**	**4**	**5**
The system operates smoothly across different windows devices.					
Cloud integration remains stable across platforms.					
Compatibility	**1**	**2**	**3**	**4**	**5**
The system integrates well with existing school databases and systems.					
Usability	**1**	**2**	**3**	**4**	**5**
The user interface is intuitive and easy to navigate.					
Instructions and labels are clear and easy to understand.					
Reliability	**1**	**2**	**3**	**4**	**5**

The system performs consistently during repeated use.					
Attendance logs are generated accurately and reliably.					
Security	**1**	**2**	**3**	**4**	**5**
User data is securely stored and protected from unauthorized access.					
The system effectively detects spoofing or fake identities.					
Maintainability	**1**	**2**	**3**	**4**	**5**
The code is modular and allows for easy updates.					
System errors can be resolved efficiently.					

APPENDIX C: SURVEY RESULT TALLY

This appendix presents the complete tabulated results of the evaluation survey conducted for the Smart Campus Security System. The data contained in this tally served as the primary basis for the statistical analysis and interpretation of the system's quality attributes against the ISO/IEC 25010 standard, as detailed in Chapter 4 of this study.

Responden	A1	A2	B1	B2	C1	C2	D1	E1	E2	F1	F2	G1	G2	H1	H2
1	4	4	4	4	4	4	4	5	5	4	4	4	4	4	3
2	4	3	4	4	2	2	3	3	4	4	4	4	5	5	5
3	3	4	3	3	3	4	3	4	4	3	4	4	3	3	3
4	5	5	5	5	4	4	5	5	5	5	5	5	5	5	5
5	4	4	4	3	3	4	5	5	4	3	4	4	4	3	4
6	5	5	5	4	4	5	5	4	4	5	4	5	5	5	5
7	4	4	4	5	4	4	4	5	5	4	5	4	5	4	5
8	2	3	3	4	3	4	3	3	4	3	2	3	3	2	2
9	4	4	5	4	5	4	4	5	5	5	5	4	5	5	5
10	4	4	4	4	4	3	3	3	5	4	5	3	3	4	3
11	3	2	4	4	4	3	3	3	4	4	4	3	3	4	3
12	4	3	4	3	3	3	4	4	3	4	5	3	4	3	3
13	4	4	4	3	3	3	3	4	4	4	4	3	3	3	3
14	4	3	4	4	4	3	3	3	4	4	4	3	3	4	3
15	4	4	4	4	3	4	5	5	5	5	5	4	4	4	3
1	5	5	5	5	4	3	4	4	4	4	5	4	5	5	4

6															
17	5	4	5	5	5	5	5	5	5	4	4	5	5	5	4
18	4	5	5	5	3	4	4	4	4	3	3	3	3	3	3
19	5	4	5	5	4	4	4	5	4	4	4	5	5	4	3
20	4	4	3	3	5	5	4	4	5	4	4	4	5	5	4
21	5	5	4	4	4	5	4	5	5	5	5	5	5	3	4
22	4	4	4	3	3	4	3	4	3	4	3	4	3	3	4
23	4	5	3	4	4	4	3	4	4	4	4	4	3	4	4
24	4	5	4	4	5	4	5	4	4	4	4	5	4	5	4
25	4	3	4	4	4	4	4	4	4	4	5	4	4	4	4
26	1	1	3	3	1	1	4	2	3	4	4	4	4	2	2
27	1	3	1	4	1	4	1	4	4	1	4	4	1	4	4
28	3	3	3	3	3	3	3	3	3	3	3	2	3	3	3
29	4	2	4	4	4	4	4	4	4	4	4	4	4	4	4
30	4	2	3	3	2	2	1	2	1	2	3	1	4	4	3
31	4	3	3	3	3	3	3	3	3	3	3	3	4	3	4	3
32	4	3	2	4	3	4	3	4	3	3	4	3	2	3	1

33	5	4	4	4	4	4	4	4	4	4	4	5	5	4	4
34	1	1	3	3	2	3	3	2	2	2	2	3	2	2	1
35	4	3	3	3	3	3	3	3	3	3	4	3	3	4	3
36	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
37	4	4	4	4	4	5	4	5	5	4	5	4	5	5	5
38	4	4	4	3	3	5	1	4	4	3	3	1	3	4	5
39	5	5	3	2	2	4	1	5	5	4	3	1	5	5	5
40	5	5	4	4	3	3	4	4	3	3	4	4	5	4	4
41	4	4	5	4	4	4	4	3	5	3	4	5	5	5	4
42	5	4	4	4	3	3	4	3	4	5	5	4	5	5	4
43	5	5	5	4	3	4	4	4	5	4	4	5	5	4	4
44	3	2	4	3	5	5	1	5	4	4	3	1	3	3	4
45	5	4	4	3	4	4	4	45	4	5	5	5	4	5	5
	3.93	**3.69**	**3.84**	**3.76**	**3.44**	**3.73**	**3.49**	**4.82**	**4.00**	**3.76**	**4.00**	**3.69**	**3.91**	**3.93**	**3.67**

APPENDIX D: LETTER

October 14, 2025

Raffy A. Melegrito
Dean, Information Technology Department
HGBaquiran College
Centro, Tumauini Isabela

Subject: Authorization to Conduct User Evaluation Survey for Final Research Project

Dear Sir,

This letter respectfully seeks your approval for BS in Information Technology students to conduct a user evaluation survey for their capstone project, **“Smart Campus Security: Facial Recognition-Based Entry and Attendance Monitoring System.”**

The survey aims to assess the system’s performance and quality based on the ISO/IEC 25010 standard, focusing on functional suitability, performance efficiency, usability, reliability, security, maintainability, and portability. The results will be used to evaluate effectiveness, identify improvements, and ensure alignment with user and institutional standards.

The students involved in this research project are:

- Cristine Joy Q. Sagadraca
- Chellsy Fei B. Manalo
- Dan Isen L. Ong
- Jesus D. Diaz

The researchers have successfully developed the “Smart Campus Security: Facial Recognition-Based Entry and Attendance Monitoring System” and are now conducting user evaluation as the final phase of their capstone study, in partial fulfillment of the BS in Information Technology degree.

All information gathered will be treated with strict confidentiality and used solely for academic purposes. Responses will remain anonymous and protected.

Your approval of this request will greatly contribute to the successful completion of the study. Thank you for your continued support.

Raffy A. Melegrito
Dean, Information Technology Department
HGBaquiran College

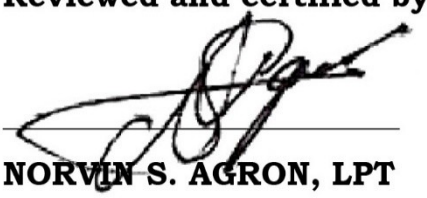
APPENDIX E: STATISTICIAN CERTIFICATION

Certificate of Statistical Review

This is to certify that the research study entitled **"Smart Campus Security: Facial Recognition-Based Entry and Attendance Monitoring System"**, conducted by **Cristine Joy Q. Sagadraca**, **Chellsy Fei B. Manalo**, **Dan Isen Lois Ong** and **Jesus Diaz** undergone a thorough review focusing on the statistical treatment of data.

Upon evaluation, I affirm that the statistical methods employed—specifically the weighted mean – are methodologically sound and appropriately aligned with the research objectives and questions. Moreover, the computations, analyses, and presentation of the statistical findings are accurate, valid, and consistent with accepted research standards.

Reviewed and certified by:



NORVIN S. AGRON, LPT
Bayabo, Delfin Albano, Isabela
Date: December 1, 2025

APPENDIX F: ENGLISH CRITIQUE CERTIFICATION

GRAMMARIAN'S CERTIFICATION

This is to certify that the undersigned has reviewed and proofread the Capstone titled: "**Smart Campus Security : Facial Recognition – Based Entry and Attendance Monitoring System"** developed by **Cristine Joy Q. Sagadraca, Jesus D. Diaz, Chellsy Fei B. Manalo, and Dan Isenlois L. Ong** based on the set of structural rules that govern the composition of sentence, phrases, and words in the English langauge. Also, all corrections and recommendations made have been incorporated into the final manuscript.

This certification is issued to the aforementioned researchers for English Critic requirements.

Issued this ___ day of _____ in the year of the Lord, _____.

ABIGAIL A. CABIA, LPT, MST

APPENDIX G: SOURCE CODE

This appendix contains selected portions of the source code for the Smart Campus Security System. The provided snippets illustrate the implementation of the system's core architecture, including the frontend interface components for the monitoring dashboard and student registration, the facial recognition API integration using OpenCV and face_recognition libraries, and the data management layer for interacting with the cloud database and handling attendance records.

Login View

```
using DevExpress.XtraBars;

using DevExpress.XtraSplashScreen;

using FRAMSx.Model;

using FRAMSx.Services;

using OFS.Lib;

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FRAMSx.View

{ public partial class LoginView : DevExpress.XtraEditors.XtraUserControl
```

```
{   private SupabaseService supabase;
    private readonly Control mainPanel;
    private readonly BarButtonItem barButtonItem;
    private readonly BarStaticItem barProfile;
    private List<Account> accounts = new List<Account>();
        private List<AccountPermission> accountPermission = new
List<AccountPermission>();
                public LoginView(Control mainPanel, BarButtonItem
barButtonItem, BarStaticItem barProfile )
    {
        InitializeComponent();
        this.mainPanel = mainPanel;
        this.barButtonItem = barButtonItem;
        this.barProfile = barProfile;
    }
    IOverlaySplashScreenHandle ShowProgressPanel()
    {
        var options = new OverlayWindowOptions(
        startupDelay: 0,
        backColor: Color.DarkGray,
        opacity: 0.5,
        fadeIn: true,
        fadeOut: true,
        imageSize: new Size(32, 32)
        );
```

```
            return SplashScreenManager.ShowOverlayForm(FindForm(),
options);
        }
        void CloseProgressPanel(IOverlaySplashScreenHandle handle)
        {
            if (handle != null)
                SplashScreenManager.CloseOverlayForm(handle);
        }
        private async Task InitializeSupabaseAndDataAsync()
        {
            if (supabase == null)
            {
                supabase = new SupabaseService();
                await supabase.InitializeAsync();
            }
            var accounstTask = supabase.GetTAsync<Account>( );
                                        var    accountPermissionTask    =
supabase.GetTAsync<AccountPermission>();
            await Task.WhenAll( accounstTask, accountPermissionTask);
            accounts = await accounstTask;
            accountPermission = await accountPermissionTask;
            if (barButtonItem != null)
            {
                barButtonItem.Visibility = BarItemVisibility.Never;
            }
```

```
            if (barProfile != null)
            {
                barProfile.Caption = "";
            }
        }
        private async Task LoadDataAndInitializeAsync()
        {
            await InitializeSupabaseAndDataAsync();
        }
                private void textEdit2_EditValueChanged(object sender,
EventArgs e)
        {
        }
        private async Task LoginAccount()
        {
            IOverlaySplashScreenHandle handle = null;
            try
            {
                handle = ShowProgressPanel();
                if (accounts.Count < 0)
                {
                    await LoadDataAndInitializeAsync();
                }
                var validation = ApplicationUtilities.ValidationInstance();
```

validation.ClearError(new Control[] { textEditEmail, textEditPassword });

var err = 0;

err += validation.IsNullOrEmpty(textEditEmail, "Email is required.", DevExpress.Utils.BeakPanelBeakLocation.Left);

err += validation.IsNullOrEmpty(textEditPassword, "Password is required.", DevExpress.Utils.BeakPanelBeakLocation.Left);

var accountValid = accounts.Where(x => x.Email.ToLower() == textEditEmail.Text.ToLower() && x.Password == textEditPassword.Text).FirstOrDefault();

if (accountValid != null)

{

// Check if account is active

if (accountValid.Status == AccountStatus.Inactive)

{

validation.SetError(textEditEmail, "Account is inactive. Please contact admin to reactive your account.");

return;

}

var permission = accountPermission.Where(x => x.RoleTypeId == accountValid.RoleTypeId).FirstOrDefault();

App.Config.AccountPermission = permission;

// Set Account Cache

App.Config.Account = accountValid;

if (barButtonItem != null)

```
                {
                    barButtonItem.Visibility = BarItemVisibility.Always;
                }
                if (barProfile != null)
                {
                        barProfile.Caption = $"{accountValid.Name}
<b><i>( {accountValid.RoleType.Role} )";
                }
                // Set UI
                  UIHelper.SetUI(mainPanel, new MenuView(permission,
mainPanel, barButtonItem, barProfile));
            }
            else
            {
                validation.SetError(textEditEmail, "Email is incorrect.");
                    validation.SetError(textEditPassword, "Password is
incorect.");
            }
        }
        catch (Exception ex)
        {
                CloseProgressPanel(handle);
            OFS.Lib.ApplicationUtilities.NotificationInstance.ShowAlert
(
                OFS.Lib.UserInterfaceManager.GetMainFormInstance,
```

```
            "Initialization Failed.",
            ex.Message,
            alertType: OFS.Lib.Utils.AlertType.Error);
        Console.WriteLine(ex.Message);
    }
    finally
    {
        CloseProgressPanel(handle);
    }
}
private async  void bntSignIn_Click(object sender, EventArgs e)
{
    await LoginAccount();
}
private async void LoginView_Load(object sender, EventArgs e)
{
    IOverlaySplashScreenHandle handle = null;
    try
    {
        handle = ShowProgressPanel();
        await LoadDataAndInitializeAsync();
    }
    catch (Exception ex)
    {
```

```
            CloseProgressPanel(handle);
             OFS.Lib.ApplicationUtilities.NotificationInstance.ShowAlert
(
                OFS.Lib.UserInterfaceManager.GetMainFormInstance,
                "Initialization Failed.",
                ex.Message,
                alertType: OFS.Lib.Utils.AlertType.Error);
            Console.WriteLine(ex.Message);
        }
        finally
        {
            CloseProgressPanel(handle);
        }
    }
     private void linkForgotPassword_Click(object sender, EventArgs
e)
    {
                                        UIHelper.SetUI(mainPanel,  new
AccountRecoveryView(mainPanel, barButtonItem, barProfile));
    }
         private async void textEditPassword_Enter(object sender,
EventArgs e)
    {
    }
```

```
        private async void textEditPassword_KeyDown(object sender,
KeyEventArgs e)
      {
         if(e.KeyCode == Keys.Enter)
         {
            await LoginAccount();
         }
      }
   }
}
```

Student View

```
using DevExpress.XtraSplashScreen;
using FRAMSx.Model;
using FRAMSx.Services;
using FRAMSx.View.Dialogs;
using FRAMSx.View.Misc;
using Supabase.Realtime.Interfaces;
using System;
using System.Collections.Generic;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Threading.Tasks;
```

```
using System.Windows.Forms;

namespace FRAMSx.View {

                    public partial class StudentView :
DevExpress.XtraEditors.XtraUserControl {

    private IRealtimeChannel _blockChannel;

    private SupabaseService supabase;

    public StudentView() {

        InitializeComponent();

        HandleDestroyed += (s, e) =>

        {

            if (_blockChannel != null) {

                _blockChannel.Unsubscribe();

                _blockChannel = null;

            }

        };

    }

    IOverlaySplashScreenHandle ShowProgressPanel() {

        var options = new OverlayWindowOptions(

        startupDelay: 0,

        backColor: Color.DarkGray,

        opacity: 0.5,

        fadeIn: true,

        fadeOut: true,

        imageSize: new Size(32, 32)
```

```
            );
              return SplashScreenManager.ShowOverlayForm(FindForm(),
options);
        }
        void CloseProgressPanel(IOverlaySplashScreenHandle handle) {
            if (handle != null)
                SplashScreenManager.CloseOverlayForm(handle);
        }
        private async Task InitializeSupabaseAndDataAsync() {
                if (supabase == null) {
                supabase = new SupabaseService();
                await supabase.InitializeAsync();
            }
            var records = await supabase.GetTAsync<Student>();
            var courses = await supabase.GetTAsync<Course>();
            GetDepartmentCount(courses, records);
           studentBindingSource.DataSource = records;
                                                    _blockChannel     =     await
supabase.SubscribeToAll<Student>(async e =>
            {
                Invoke((MethodInvoker)(async () =>
                {
                    var updated = await supabase.GetTAsync<Student>();
                    studentBindingSource.DataSource = updated;
                    studentBindingSource.ResetBindings(false);
```

```
                GetDepartmentCount(courses, updated);
            }));
        });
    }
        private void GetDepartmentCount(List<Course> courses, List<Student> students) {
        flowLayoutTiles.Controls.Clear();
        foreach (var item in courses) {
            var newTile = new Misc.TileItem(students.Where(x => x.CourseId == item.Id).Count().ToString() ?? "0", item.ShortName, Color.RoyalBlue, TileType.Student);
            flowLayoutTiles.Controls.Add(newTile);
        }
    }
    private async Task LoadDataAndInitializeAsync() {
      await InitializeSupabaseAndDataAsync();
    }
     private async void StudentView_Load(object sender, EventArgs e) {
        IOverlaySplashScreenHandle handle = null;
        try {
            handle = ShowProgressPanel();
            await LoadDataAndInitializeAsync();
        }
        catch (Exception ex) {
```

```
            CloseProgressPanel(handle);
             OFS.Lib.ApplicationUtilities.NotificationInstance.ShowAlert
(
                OFS.Lib.UserInterfaceManager.GetMainFormInstance,
                "Initialization Failed.",
                ex.Message,
                alertType: OFS.Lib.Utils.AlertType.Error);
            Console.WriteLine(ex.Message);
        }
        finally {
            CloseProgressPanel(handle);
        }
    }
        private void repositoryItemButtonEdit1_ButtonClick(object
sender, DevExpress.XtraEditors.Controls.ButtonPressedEventArgs e)
{
        if(e.Button.Index == 0) {
            // Edit Student
            if (gridView1.GetFocusedRow() is Student student) {
                                                UIHelper.ShowUCDialog(new
StudentDetailView(student), title: "Student Details");
            }
        }
        if (e.Button.Index == 1) {
            // Edit Student
```

```
            if (gridView1.GetFocusedRow() is Student student)
            {
                                                UIHelper.ShowUCDialog(new
UserAttendanceDataView(student: student), title: "Student
Attendance Log");
            }
        }
    }
            private void navButton3_ElementClick(object sender,
DevExpress.XtraBars.Navigation.NavElementEventArgs e)
    {
        UIHelper.ShowUCDialog(new StudentRegistrationView(), title:
"Student Registration");
    }
  }
}
```

Student Registration Form

```
using System;
using System.ComponentModel;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
using DevExpress.XtraGrid.Views.Tile;
using DevExpress.XtraSplashScreen;
using Emgu.CV;
using Emgu.CV.Structure;
using FRAMSx.Model;
using FRAMSx.Services;
using OFS.Lib;
namespace FRAMSx.View
{
        public partial class StudentRegistrationView : DevExpress.XtraEditors.XtraUserControl
    {
        private VideoCapture _capture;
        private CascadeClassifier _faceCascade;
        private BindingList<ImageDataset> imageDatasets;
        private SupabaseService supabase;
                private readonly string haarCascadeFile = Path.Combine(Application.StartupPath,
"haarcascade_frontalface_default.xml");
                private readonly string datasetPath = Path.Combine(Application.StartupPath, "dataset");
        private volatile bool _isProcessingFrame = false;
        public StudentRegistrationView()
        {
            InitializeComponent();
```

```
        }
            private async void StudentRegistrationView_Load(object
sender, EventArgs e)
        {
            IOverlaySplashScreenHandle handle = null;
           try
           {
              handle = ShowProgressPanel();
              await LoadDataAndInitializeAsync();
           }
           catch (Exception ex)
           {
              CloseProgressPanel(handle);
               OFS.Lib.ApplicationUtilities.NotificationInstance.ShowAlert
(
                  OFS.Lib.UserInterfaceManager.GetMainFormInstance,
                  "Initialization Failed.",
                  ex.Message,
                  alertType: OFS.Lib.Utils.AlertType.Error);
              Console.WriteLine(ex.Message);
           }
           finally
           {
              CloseProgressPanel(handle);
           }
```

```
}
private async Task LoadDataAndInitializeAsync()
{
    await InitializeSupabaseAndDataAsync();
    await Task.Run(() =>
    {
        _capture = new VideoCapture(0);
        _faceCascade = new CascadeClassifier(haarCascadeFile);
    });
    imageDatasets = new BindingList<ImageDataset>();
    imageDatasetBindingSource.DataSource = imageDatasets;
    timer1.Start();
}
private async Task InitializeSupabaseAndDataAsync()
{
    if (supabase == null)
    {
        supabase = new SupabaseService();
        await supabase.InitializeAsync();
    }
    var coursesTask = supabase.GetTAsync<Course>();
    var blocksTask = supabase.GetTAsync<Block>();
    var yearLevelsTask = supabase.GetTAsync<YearLevel>();
```

```
                    await Task.WhenAll(coursesTask, blocksTask,
yearLevelsTask);
            courseBindingSource.DataSource = await coursesTask;
            blockBindingSource.DataSource = await blocksTask;
            yearLevelBindingSource.DataSource = await yearLevelsTask;
            comboBoxCourse.ItemIndex = 0;
            comboBoxBlock.ItemIndex = 0;
            comboBoxYearLevel.ItemIndex = 0;
        }
        IOverlaySplashScreenHandle ShowProgressPanel()
        {
            var options = new OverlayWindowOptions(
            startupDelay: 0,
            backColor: Color.DarkGray,
            opacity: 0.5,
            fadeIn: true,
            fadeOut: true,
            imageSize: new Size(32, 32)
            );
              return SplashScreenManager.ShowOverlayForm(FindForm(),
options);
        }
        void CloseProgressPanel(IOverlaySplashScreenHandle handle)
        {
            if (handle != null)
```

```
            SplashScreenManager.CloseOverlayForm(handle);
        }
        private void timer1_Tick(object sender, EventArgs e)
        {
        // If the previous frame is still processing, skip this tick to avoid backlog.
          if (_isProcessingFrame || _capture == null) return;
           try {
              _isProcessingFrame = true;
              using (var frame = _capture.QueryFrame()) {
                 if (frame != null) {
                    using (var img = frame.ToImage<Bgr, byte>()) {
                       var gray = img.Convert<Gray, byte>();
                         var faces = _faceCascade.DetectMultiScale(gray, 1.1, 10, Size.Empty, Size.Empty);
                       foreach (var face in faces) {
                          img.Draw(face, new Bgr(Color.Green), 2);
                       }
                      imageBoxEmguCV.Image = img;
                    }
                 }
              }
           }
           finally {
              // Allow the next frame to be processed.
```

```
            _isProcessingFrame = false;
        }
    }
    private void btnCapture_Click(object sender, EventArgs e)
    {
     var uid = Guid.NewGuid();
        using (var frame = _capture.QueryFrame())
        {
            if (frame != null)
            {
                using (var img = frame.ToImage<Bgr, byte>())
                {
                    var gray = img.Convert<Gray, byte>();
                    var faces = _faceCascade.DetectMultiScale(gray, 1.1,
10, Size.Empty, Size.Empty);
                    foreach (var face in faces)
                    {
                        var faceImg = img.GetSubRect(face);
                            //var resizedFace = faceImg.Resize(200, 200,
Emgu.CV.CvEnum.Inter.Linear);
                        imageDatasets.Add(new ImageDataset()
                        {
                            Capture = faceImg.ToBitmap(),
                            Label = uid.ToString()
                        });
```

```
                    break;
                }
            }
        }
    }
}
private void ResetFields()
{
    textStudentID.Text = "";
    textFullName.Text = "";
    textAddress.Text = "";
    if (imageDatasets != null && imageDatasets.Count > 0)
    {
        imageDatasetBindingSource.Clear();
    }
}
private void btnReset_Click(object sender, EventArgs e)
{
    ResetFields();
}
private async void btnSave_Click(object sender, EventArgs e)
{
    // Validate
    var validate = ApplicationUtilities.ValidationInstance();
```

```
                validate.ClearError(new Control[] {gridControl1, textStudentID, textFullName, textAddress });
            var err = 0;
               err += validate.IsNullOrEmpty(textStudentID, "Student ID cannot be empty.", DevExpress.Utils.BeakPanelBeakLocation.Left);
                err += validate.IsNullOrEmpty(textFullName, "Full name cannot be empty.", DevExpress.Utils.BeakPanelBeakLocation.Left);
             err += validate.IsNullOrEmpty(textAddress, "Address cannot be empty.", DevExpress.Utils.BeakPanelBeakLocation.Left);
            // Chcek if there are image dataset
            if(imageDatasets.Count == 0)
            {
                validate.SetError(gridControl1, "Student image dataset is required.");
               err++;
            }
            if (err > 0)
            {
               return;
            }
            // Save Information to DB First
            var student = new StudentInsert()
            {
               StudentId = textStudentID.EditValue.ToString(),
               Name = textFullName.Text,
```

```
            Address = textAddress.Text,
            BlockId = Convert.ToInt32(comboBoxBlock.EditValue),
            CourseId = Convert.ToInt32(comboBoxCourse.EditValue),
                                                    YearLevelId     =
Convert.ToInt32(comboBoxYearLevel.EditValue)
        };
        try
        {           SplashScreenManager.ShowForm(typeof(View.Misc.
WaitDialog));
            // Trap duplicate Student Id
            var students = await supabase.GetTAsync<Student>();
              var checkStudentId = students.Where(x => x.StudentId
== textStudentID.EditValue.ToString()).FirstOrDefault();
            if(checkStudentId != null)
            {
                  validate.SetError(textStudentID, "Student Id already
registered.");
                return;
            }
            // Insert Student Data
            var result = await supabase.InsertTAsync(student);
            if (result[0] != null) {
                // Insert Image Dataset
                 var datasetsToInsert = imageDatasets.Select(item =>
new DatasetInsert
```

```
                {
                    FileName = item.Label,
                    StudentEmployeeID = textStudentID.Text,
                    DatasetType = DatasetType.Student
                }).ToList();
                foreach (var item in datasetsToInsert)
                {
                    await supabase.InsertTAsync(item);
                }
            }
            // Save Images to the dataset folder
            foreach(var item in imageDatasets)
            {
                var fileName = $"{item.Label}.jpg";
                        item.Capture.Save(Path.Combine(datasetPath,
fileName));
            }
            ApplicationUtilities.NotificationInstance.ShowAlert(
              UserInterfaceManager.GetMainFormInstance,
              "Success!",
              "Student has been registered",
              alertType: OFS.Lib.Utils.AlertType.Success);
            ResetFields();
        }
```

```
            catch (Exception ex)
            {
                ApplicationUtilities.NotificationInstance.ShowAlert(
                    UserInterfaceManager.GetMainFormInstance,
                    "Something went wrong.",
                    ex.Message,
                    alertType: OFS.Lib.Utils.AlertType.Error);
                Console.WriteLine(ex.Message);
            }
            finally
            {
                SplashScreenManager.CloseForm(false);
            }
        }
                    private void tileView1_ItemClick(object sender,
DevExpress.XtraGrid.Views.Tile.TileViewItemClickEventArgs e)
        {
            var deleteItem = sender as TileView;
                var id = deleteItem.GetRowCellValue(e.Item.RowHandle,
"Label");
            if (id != null) {
                                                UIHelper.ShowUCDialog(new
View.Dialogs.DeleteDatasetImage(imageDatasets, id.ToString()));
            }
        }
```

```
    }
}
```

Monitoring View

```
using DevExpress.XtraEditors;
using DevExpress.XtraSplashScreen;
using Emgu.CV;
using Emgu.CV.Face;
using Emgu.CV.Structure;
using FRAMSx.Model;
using FRAMSx.Properties;
using FRAMSx.Services;
using Supabase.Realtime.Interfaces;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Management;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace FRAMSx.View
```

```
{
    public partial class MonitoringView : UserControl
    {
        private IRealtimeChannel _attendanceChannel;
        private VideoCapture _capture;
        private CascadeClassifier _faceCascade;
        private LBPHFaceRecognizer recognizer;
        private Emgu.CV.UI.ImageBox imageBox;
                        private readonly string haarCascadeFile =
Path.Combine(Application.StartupPath,
"haarcascade_frontalface_default.xml");
        private int timeInInterval = Settings.Default.TimeInInterval;
                                    private int confidencLevel =
Settings.Default.RecognitionConfidence;
                private List<DatasetMapper> datasetMappers = new
List<DatasetMapper>();
        private List<Student> students = new List<Student>();
        private List<Course> courses = new List<Course>();
        private List<Block> blocks = new List<Block>();
        private List<YearLevel> yearLevels = new List<YearLevel>();
        private List<Employee> employees = new List<Employee>();
        private List<Position> positions = new List<Position>();
                    private List<Attendance> attendances = new
List<Attendance>();
        private SupabaseService supabase;
```

```
private bool _isProcessing = false;

public MonitoringView()
{
    InitializeComponent();
    HandleDestroyed += (s, e) =>
    {
        if (_attendanceChannel != null)
        {
            _attendanceChannel.Unsubscribe();
            _attendanceChannel = null;
        }
    };
}
IOverlaySplashScreenHandle ShowProgressPanel()
{
    var options = new OverlayWindowOptions(
    startupDelay: 0,
    backColor: Color.DarkGray,
    opacity: 0.8,
    fadeIn: true,
    fadeOut: true,
    imageSize: new Size(32, 32)
    );
```

```
            return SplashScreenManager.ShowOverlayForm(FindForm(),
options);
        }
        void CloseProgressPanel(IOverlaySplashScreenHandle handle)
        {
            if (handle != null)
                SplashScreenManager.CloseOverlayForm(handle);
        }
        private List<Camera> GetAllConnectedCamera()
        {
            var cameraNames = new List<Camera>();
                                        using (var searcher = new
ManagementObjectSearcher("SELECT * FROM Win32_PnPEntity
WHERE (PNPClass = 'Image' OR PNPClass = 'Camera')"))
            {
                var count = 0;
                foreach (var device in searcher.Get())
                {
                                cameraNames.Add(new Camera() { Name =
device["Caption"].ToString(), Index = count++ });
                }
            }
            return cameraNames;
        }
        private async Task InitializeSupabaseAndDataAsync()
```

```
{
    if (supabase == null)
    {
        supabase = new SupabaseService();
        await supabase.InitializeAsync();
    }
                    var cameraName = GetAllConnectedCamera()
[Settings.Default.CameraIndex].Name;
                     lblSelectedCamera.Text = $"Online Camera:
{cameraName}";
                                                _capture = new
VideoCapture(Settings.Default.CameraIndex);
    _faceCascade = new CascadeClassifier(haarCascadeFile);
      recognizer = new LBPHFaceRecognizer(1, 8, 8, 8, 100); // 
Match training parameters
    // Set Reading Speed in ms
    timer1.Interval = Settings.Default.TimeInInterval;
         var modelPath = Path.Combine(Application.StartupPath,
"model", "trained_model.yml");
    if (File.Exists(modelPath))
    {
        recognizer.Read(modelPath);
    }
    else
    {
```

```
            Console.WriteLine("Trained model not found. Recognition
will fail.");

                //XtraMessageBox.Show("Trained model not found.
Recognition will fail.", "No model detected.",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

            OFS.Lib.ApplicationUtilities.NotificationInstance.ShowAlert
(

          OFS.Lib.UserInterfaceManager.GetMainFormInstance,

          "No model detected.",

         "Trained model not found. Recognition will fail.",

          alertType: OFS.Lib.Utils.AlertType.Error);

           return;

        }

                                        var attendanceTask =
supabase.GetTAsync<Attendance>("date",
DateTime.Now.ToString("yyyy-MM-dd"), ascending: false);

                                        var datasetMappersTask =
supabase.GetTAsync<DatasetMapper>();

        var studentsTask = supabase.GetTAsync<Student>();

        var coursesTask = supabase.GetTAsync<Course>();

        var blocksTask = supabase.GetTAsync<Block>();

        var yearLevelsTask = supabase.GetTAsync<YearLevel>();

        var employeesTask = supabase.GetTAsync<Employee>();

            await Task.WhenAll(datasetMappersTask, studentsTask,
coursesTask, blocksTask, yearLevelsTask, employeesTask,
attendanceTask);
```

```
            datasetMappers = await datasetMappersTask;
            students = await studentsTask;
            courses = await coursesTask;
            blocks = await blocksTask;
            yearLevels = await yearLevelsTask;
            employees = await employeesTask;
            attendances = await attendanceTask;
           attendanceBindingSource.DataSource = attendances;
                                                _attendanceChannel = await
supabase.SubscribeToAll<Attendance>(e =>
            {
               Invoke((MethodInvoker)(async () =>
               {
                                                    var updated = await
supabase.GetTAsync<Attendance>("date",
DateTime.Now.ToString("yyyy-MM-dd"), ascending: false);
                  attendanceBindingSource.DataSource = null;
                  attendanceBindingSource.ResetBindings(false);
                  attendanceBindingSource.DataSource = updated;
                  attendances = updated;
                                Console.WriteLine($"Updated Record Count:
{updated.Count}");
               }));
            });
            // If dataset Mapper is empty do not proceed
```

```
            if (datasetMappers == null || datasetMappers.Count == 0)
            {
                XtraMessageBox.Show("There are no image dataset saved. Please register first and train the model.", "No image dataset.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                btnStartMonitoring.Enabled = false;
            }
        }
        private async Task LoadDataAndInitializeAsync()
        {
            // Network calls are naturally async, so we just await them.
            await InitializeSupabaseAndDataAsync();
        }
        private async void MonitoringView_Load(object sender, EventArgs e)
        {
            IOverlaySplashScreenHandle handle = null;
            try
            {
                handle = ShowProgressPanel();
                panelCameraFeed.Controls.Clear();
                panelCameraFeed.Controls.Add(new View.Misc.MonitoringPlaceholderView() { Dock = DockStyle.Fill });
                await LoadDataAndInitializeAsync();
            }
```

```
            catch (Exception ex)
            {
                // It's good practice to close the panel even if an error
occurs.
                CloseProgressPanel(handle);
                OFS.Lib.ApplicationUtilities.NotificationInstance.ShowAlert
(
                    OFS.Lib.UserInterfaceManager.GetMainFormInstance,
                    "Initialization Failed.",
                    ex.Message,
                    alertType: OFS.Lib.Utils.AlertType.Error);
                Console.WriteLine(ex.Message);
            }
            finally
            {
                CloseProgressPanel(handle);
            }
        }
        private void switchBg(string label)
        {
            if (label == "error")
            {
                panelRecognitionStatus.BackColor = Color.DarkRed;
            }
            if (label == "ok")
```

```
        {
            panelRecognitionStatus.BackColor = Color.DarkGreen;
        }
    }
    private void ResetStudentDetails()
    {
        labelStudentId.Text = "";
        labelName.Text = "";
        labelCourse.Text = "";
        labelBlock.Text = "";
    }
    private async Task LogStudentAttendance(Student student)
    {
        if (_isProcessing) return;
        try
        {
            _isProcessing = true;
                    var (hasAttended, todaysAttendance) = await
supabase.CheckAttendanceTodayAsync(student.StudentId);
            if (hasAttended)
            {
                Console.WriteLine("=== HAS RECORDS TODAY ===");
                                                var lastTimeOut =
todaysAttendance.OrderByDescending(x                                =>
x.TimeStamp).FirstOrDefault();
```

```
                    //var underRestTime = DateTime.Now.TimeOfDay -
lastTimeOut.TimeStamp;

                var recentTimeIn = lastTimeOut.TimeStamp;

               var currTime = DateTime.Now.TimeOfDay;

                        var isOnRest = (currTime - recentTimeIn) >
TimeSpan.FromSeconds(timeInInterval);

                if (isOnRest)

                {

                   // 1 minute has passed

                                        if (lastTimeOut.StatusType ==
AttendanceStatusType.TimeIn)

                    {

                       Console.WriteLine("*** Record Time Out");

                                            var newAttendance =
SetAttendanceLog(student.Name,              student.StudentId,
student.CourseId,                       AccountType.Student,
AttendanceStatusType.TimeOut);

                                           var hasInsert = await
supabase.InsertTAsync(newAttendance);

                       //if (hasInsert.Count > 0) {

                            //     Console.WriteLine("=== DATA INSERTED
===");

                       //}

                    }

                    else

                    {
```

```
                Console.WriteLine("*** Record Time In");
                var newAttendance = SetAttendanceLog(student.Name, student.StudentId, student.CourseId, AccountType.Student, AttendanceStatusType.TimeIn);
                var hasInsert = await supabase.InsertTAsync(newAttendance);
            }
        }
        else
        {
            Console.WriteLine("=== UNDER REST ===");
        }
    }
    else
    {
        Console.WriteLine("=== NO DATA ===");
        var newAttendance = SetAttendanceLog(student.Name, student.StudentId, student.CourseId, AccountType.Student, AttendanceStatusType.TimeIn);
        var hasInsert = await supabase.InsertTAsync(newAttendance);
        if (hasInsert.Count > 0)
        {
            Console.WriteLine("=== DATA INSERTED ===");
        }
```

```
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
        finally
        {
            _isProcessing = false;
        }
    }
        private async Task LogEmployeeAttendance(Employee employee)
    {
        if (_isProcessing) return;
        try
        {
            _isProcessing = true;
                    var (hasAttended, todaysAttendance) = await supabase.CheckAttendanceTodayAsync(employee.EployeeId);
            if (hasAttended)
            {
                                                var lastTimeOut = todaysAttendance.OrderByDescending(x => x.TimeStamp).FirstOrDefault();
```

```
                    //var underRestTime = DateTime.Now.TimeOfDay -
lastTimeOut.TimeStamp;

                var recentTimeIn = lastTimeOut.TimeStamp;

                var currTime = DateTime.Now.TimeOfDay;

                        var isOnRest = (currTime - recentTimeIn) >
TimeSpan.FromSeconds(timeInInterval);

               if (isOnRest)

                {

                   // 1 minute has passed

                                        if (lastTimeOut.StatusType ==
AttendanceStatusType.TimeIn)

                    {

                                                var newAttendance =
SetAttendanceLog(employee.Name,          employee.EployeeId,
employee.PositionId,                     AccountType.Employee,
AttendanceStatusType.TimeOut);

                                               var hasInsert = await
supabase.InsertTAsync(newAttendance);

                    }

                    else

                    {

                                                var newAttendance =
SetAttendanceLog(employee.Name,          employee.EployeeId,
employee.PositionId,                     AccountType.Employee,
AttendanceStatusType.TimeIn);

                                               var hasInsert = await
supabase.InsertTAsync(newAttendance);
```

```
                }
            }
        }
        else
        {
                                    var newAttendance = SetAttendanceLog(employee.Name, employee.EployeeId, employee.PositionId, AccountType.Employee, AttendanceStatusType.TimeIn);
                                    var hasInsert = await supabase.InsertTAsync(newAttendance);
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"An error occurred: {ex.Message}");
    }
    finally
    {
        _isProcessing = false;
    }
}
    private AttendanceInsert SetAttendanceLog(string name, string accountId, int posCourseId, AccountType accountType, AttendanceStatusType attendanceStatusType)
    {
```

```
            return new AttendanceInsert() { Name = name, AccountId =
accountId, PosCourseId = posCourseId, AccountType = accountType,
StatusType = attendanceStatusType, Date =
DateTime.Today.ToString("yyyy-MM-dd"), TimeStamp =
DateTime.Now.TimeOfDay };
        }
        private async Task GetDetails(int index)
        {
            try
            {
                var mapper = datasetMappers.Where(x => x.Label ==
index).FirstOrDefault();
                if (mapper == null)
                {
                    //return null;
                    ResetStudentDetails();
                    return;
                }
                // Student
                if (mapper.DatasetType == DatasetType.Student)
                {
                    var student = students.Where(x => x.StudentId ==
mapper.AccountId).FirstOrDefault();
                    if (student == null) return;
                    labelStudentId.Text = student.StudentId;
                    labelName.Text = student.Name;
```

```
                labelCourse.Text = student.Course.ShortName;
                labelBlock.Text = student.Block.ShortName;
                // logging Attendance
                await LogStudentAttendance(student);
            }
            if (mapper.DatasetType == DatasetType.Employee)
            {
                var employee = employees.Where(x => x.EployeeId
== mapper.AccountId).FirstOrDefault();
                if (employee == null) return;
                labelStudentId.Text = employee.EployeeId;
                labelName.Text = employee.Name;
                labelCourse.Text = employee.Position.Name;
                // logging employee Attendance
                await LogEmployeeAttendance(employee);
            }
        }
        catch (Exception ex)
        {
            OFS.Lib.ApplicationUtilities.NotificationInstance.ShowAlert
(
                OFS.Lib.UserInterfaceManager.GetMainFormInstance,
                "Something went wrong.",
                ex.Message,
                alertType: OFS.Lib.Utils.AlertType.Error);
```

```
            Console.WriteLine(ex.Message);
        }
    }
    private string GetLabelStringName(int index)
    {
            var mapper = datasetMappers.Where(x => x.Label == index).FirstOrDefault();
        if (mapper == null)
        {
            //return null;
            ResetStudentDetails();
            return string.Empty;
        }
        if (mapper.DatasetType == DatasetType.Student)
        {
                var student = students.Where(x => x.StudentId == mapper.AccountId).FirstOrDefault();
            if (student == null) { return string.Empty; }
            return student.Name;
        }
        if (mapper.DatasetType == DatasetType.Employee)
        {
            var employee = employees.Where(x => x.EployeeId == mapper.AccountId).FirstOrDefault();
            if (employee == null) { return string.Empty; }
```

```
            return employee.Name;
        }
        return string.Empty;
    }
    private async void timer1_Tick(object sender, EventArgs e)
    {
        using (var frame = _capture.QueryFrame())
        {
            if (frame != null)
            {
                using (var img = frame.ToImage<Bgr, byte>())
                {
                    var gray = img.Convert<Gray, byte>();
                    var faces = _faceCascade.DetectMultiScale(gray, 1.1,
10, Size.Empty, Size.Empty);
                    foreach (var face in faces)
                    {
                        string labelText;
                            var faceImg = gray.Copy(face).Resize(100, 100,
Emgu.CV.CvEnum.Inter.Cubic);
                        // Crop from color for saving/display
                            var faceImgColor = img.Copy(face).Resize(100,
100, Emgu.CV.CvEnum.Inter.Cubic);
                        if (recognizer == null)
                        {
```

```
                Console.WriteLine("Recognizer not initialized.");
                return;
            }
            // Predict identity
            var result = recognizer.Predict(faceImg);
                    if (result.Label != -1 && result.Distance > Settings.Default.RecognitionConfidence)
            {
              labelControlStatus.Text = "REGISTERED";
                labelText = GetLabelStringName(result.Label);
                labelText = $"ID: {result.Label}";
                                    //Console.WriteLine($"Name: \t{GetLabelStringName(result.Label)}");
                await GetDetails(result.Label);
                // Log Attendance
              pictureBox2.Image = faceImgColor.ToBitmap();
                switchBg("ok");
            }
            else
            {
                labelControlStatus.Text = "UNKNOWN";
                labelText = "Unkown";
                ResetStudentDetails();
                pictureBox2.Image = faceImgColor.ToBitmap();
                switchBg("error");
```

```
                    }

                    img.Draw(face, new Bgr(Color.Green), 2);

                    CvInvoke.PutText(img, labelText, new Point(face.X, face.Y - 10), Emgu.CV.CvEnum.FontFace.HersheyPlain, 0.8, new Bgr(Color.Yellow).MCvScalar);

                }

                imageBox.Image = img;

            }

        }

    }

    private void btnStartMonitoring_Click(object sender, EventArgs e)

    {

        var modelPath = Path.Combine(Application.StartupPath, "model", "trained_model.yml");

        if (!File.Exists(modelPath))

        {

            Console.WriteLine("Trained model not found. Recognition will fail.");

            //XtraMessageBox.Show("Trained model not found. Recognition will fail.", "No model detected.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

            OFS.Lib.ApplicationUtilities.NotificationInstance.ShowAlert(

            OFS.Lib.UserInterfaceManager.GetMainFormInstance,
```

```
                "No model detected.",
                "Trained model not found. Recognition will fail.",
                alertType: OFS.Lib.Utils.AlertType.Error);
            return;
        }
        if (btnStartMonitoring.Text == "Start Monitoring")
        {
            //cameraFeed = new PictureBox(){ Dock = DockStyle.Fill, SizeMode = PictureBoxSizeMode.CenterImage , BackColor = Color.Gray};
                imageBox = new Emgu.CV.UI.ImageBox() { Dock = DockStyle.Fill, SizeMode = PictureBoxSizeMode.CenterImage, BackColor = Color.Gray };
            panelCameraFeed.Controls.Clear();
            panelCameraFeed.Controls.Add(imageBox);
            timer1.Start();
            btnStartMonitoring.Text = "Stop Monitoring";
                                btnStartMonitoring.BackColor = Color.FromName("@Danger");
        }
        else
        {
            timer1.Stop();
            //cameraFeed.Dispose();
            imageBox.Dispose();
```

```
            panelCameraFeed.Controls.Clear();
                                        panelCameraFeed.Controls.Add(new
View.Misc.MonitoringPlaceholderView() { Dock = DockStyle.Fill });
            btnStartMonitoring.Text = "Start Monitoring";
                                            btnStartMonitoring.BackColor =
Color.FromName("@Primary");
        }
    }
                private void tileBarItem6_ItemClick(object sender,
DevExpress.XtraEditors.TileItemEventArgs e)
    {
    }
  }
}
```

Face Recognition Configuration View

```
using DevExpress.XtraSplashScreen;
using Emgu.CV;
using Emgu.CV.Face;
using Emgu.CV.Structure;
using Emgu.CV.Util;
using FRAMSx.Model;
using FRAMSx.Properties;
using FRAMSx.Services;
using OFS.Lib;
```

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Management;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace FRAMSx.View.Data
{
        public partial class FaceRecognitionModelView : DevExpress.XtraEditors.XtraUserControl
    {
        private SupabaseService supabase;
                    private readonly string modelPath = Path.Combine(Application.StartupPath, "model", "trained_model.yml");
                    private readonly string datasetPath = Path.Combine(Application.StartupPath, "dataset");
        public FaceRecognitionModelView()
        {
            InitializeComponent();
                }
    IOverlaySplashScreenHandle ShowProgressPanel()
```

```
        {
            var options = new OverlayWindowOptions(
            startupDelay: 0,
            backColor: Color.DarkGray,
            opacity: 0.5,
            fadeIn: true,
            fadeOut: true,
            imageSize: new Size(32, 32)
            );
              return SplashScreenManager.ShowOverlayForm(FindForm(),
options);
        }
        void CloseProgressPanel(IOverlaySplashScreenHandle handle)
        {
            if (handle != null)
                SplashScreenManager.CloseOverlayForm(handle);
        }
        private async Task InitializeSupabaseAndDataAsync()
        {
            if (supabase == null)
            {
                supabase = new SupabaseService();
                await supabase.InitializeAsync();
            }
```

```
                                        var datasetMapperTask =
supabase.GetTAsync<DatasetMapper>();
            var studentsTask = supabase.GetTAsync<Student>();
            var employeesTask = supabase.GetTAsync<Employee>();
                    await Task.WhenAll(datasetMapperTask, studentsTask,
employeesTask);
            var datasetMapper =await datasetMapperTask;
            var students = await studentsTask;
            var employees = await employeesTask;
              var recentTraining = datasetMapper.OrderByDescending(x
=> x.CreatedAt).FirstOrDefault();
            if (recentTraining != null)
            {
                    labelLastTraining.Text = $"{recentTraining.CreatedAt:
MMMM dd, yyyy} at {recentTraining.CreatedAt: HH:mm tt}";
            }
            labelFacesCount.Text = $"{datasetMapper.Count} Faces";
                                        labelConfidenceThreshold.Text =
$"{Settings.Default.RecognitionConfidence}%";
                                        trackBarControl1.EditValue =
Settings.Default.RecognitionConfidence;
                                    cameraBindingSource.DataSource =
GetAllConnectedCamera();
                                        lookUpEditCameras.ItemIndex =
Settings.Default.CameraIndex;
```

```
                                        trackBarTimeInInterval.EditValue =
Settings.Default.TimeInInterval;
                                        trackBarreadingSpeed.EditValue =
Settings.Default.ReadingSpeed;
        }
      private async Task LoadDataAndInitializeAsync()
      {
         // Network calls are naturally async, so we just await them.
         await InitializeSupabaseAndDataAsync();
      }
          private async void FaceRecognitionModelView_Load(object
sender, EventArgs e)
      {
         IOverlaySplashScreenHandle handle = null;
         try
         {
            handle = ShowProgressPanel();
            await LoadDataAndInitializeAsync();
         }
         catch (Exception ex)
         {
           CloseProgressPanel(handle);
             OFS.Lib.ApplicationUtilities.NotificationInstance.ShowAlert
(
               OFS.Lib.UserInterfaceManager.GetMainFormInstance,
```

```
                "Initialization Failed.",
                ex.Message,
                alertType: OFS.Lib.Utils.AlertType.Error);
            Console.WriteLine(ex.Message);
        }
        finally
        {
            CloseProgressPanel(handle);
        }
    }
        private async void btnRetrainModel_Click(object sender,
EventArgs e)
    {
        try
        {
  SplashScreenManager.ShowForm(typeof(View.Misc.WaitDialog));
            // Get All Dataset
            var datasets = await supabase.GetTAsync<Dataset>();
            if (datasets == null || !datasets.Any()) {
                ApplicationUtilities.NotificationInstance.ShowAlert(
              UserInterfaceManager.GetMainFormInstance,
              "Something went wrong.",
              "No datasets found in Supabase.",
              alertType: OFS.Lib.Utils.AlertType.Error);
```

```
            return;
        }
        // Delete Dataset Mapper
        await supabase.DeleteDatasetMapperAsync();
        var datasetMapper = new List<DatasetMapper>();
       var recognizer = new LBPHFaceRecognizer(1, 8, 8, 8, 100);
        var images = new List<Image<Gray, byte>>();
        var labels = new List<int>();
        int count = 0;
        foreach (var dataset in datasets)
        {
            var imgName = $"{dataset.FileName}.jpg";
                    var imgFilePath = Path.Combine(datasetPath,
imgName);
            if (!File.Exists(imgFilePath))
            {
                        Console.WriteLine($"Image not found:
{imgFilePath}");
              continue;
            }
            var image = new Image<Gray, byte>(imgFilePath);
             if (image == null || image.Width == 0 || image.Height
== 0)
            {
              Console.WriteLine($"Invalid image: {imgFilePath}");
```

```
                continue;
            }
            count++;
            images.Add(image);
            labels.Add(count);
            datasetMapper.Add(new DatasetMapper
            {
                Label = count,
                FileName = imgName,
                AccountId = dataset.StudentEmployeeID,
                DatasetType = dataset.DatasetType,
                CreatedAt = DateTime.Now
            });
        }
        if (images.Count == 0 || labels.Count == 0)
            throw new InvalidOperationException("No valid training data found.");
        var trainImages = new VectorOfMat();
        foreach (var img in images)
        {
            trainImages.Push(new[] { img.Mat });
        }
        var trainLabels = new VectorOfInt(labels.ToArray());
        recognizer.Train(trainImages, trainLabels);
```

```
            recognizer.Write(modelPath);
            if (!File.Exists(modelPath))
                throw new IOException("Model file was not saved.");
                var datasetMapperItems = datasetMapper.Select(x =>
new DatasetMapper
            {
                Label = x.Label,
                FileName = x.FileName,
                AccountId = x.AccountId,
                DatasetType = x.DatasetType,
                CreatedAt = x.CreatedAt
            }).ToList();
            foreach (var item in datasetMapperItems)
            {
                await supabase.InsertTAsync(item);
Console.WriteLine($"{item.AccountId}\
t{item.DatasetType.ToString()}");
            }
            ApplicationUtilities.NotificationInstance.ShowAlert(
              UserInterfaceManager.GetMainFormInstance,
              "Training Complete",
                    $"Image training completed with {images.Count}
images.",
              alertType: OFS.Lib.Utils.AlertType.Info);
        }
```

```
            catch (Emgu.CV.Util.CvException cvEx)
            { ApplicationUtilities.NotificationInstance.ShowAlert(
                  UserInterfaceManager.GetMainFormInstance,
                  "OpenCV Error",
                  cvEx.Message,
                  alertType: OFS.Lib.Utils.AlertType.Error);
               Console.WriteLine("OpenCV error: " + cvEx.Message);
            }
            catch (Exception ex)
            {
               ApplicationUtilities.NotificationInstance.ShowAlert(
                  UserInterfaceManager.GetMainFormInstance,
                  "Something went wrong.",
                  ex.Message,
                  alertType: OFS.Lib.Utils.AlertType.Error);
               Console.WriteLine(ex.Message);
            }
            finally
            {
               SplashScreenManager.CloseForm(false);
            }
    }    private void btnSaveRecognitionSettings_Click(object sender,
EventArgs e)
      {
```

```
                                        Settings.Default.CameraIndex =
Convert.ToInt32(lookUpEditCameras.EditValue);

                                    Settings.Default.RecognitionConfidence =
Convert.ToInt32(trackBarControl1.EditValue);

                                        Settings.Default.TimeInInterval =
Convert.ToInt32(trackBarTimeInInterval.Value);

                                        Settings.Default.ReadingSpeed =
Convert.ToInt32(trackBarreadingSpeed.Value);

            Settings.Default.Save();

                                        labelConfidenceThreshold.Text =
$"{Settings.Default.RecognitionConfidence}%";

            ApplicationUtilities.NotificationInstance.ShowAlert(

                UserInterfaceManager.GetMainFormInstance,

                "Settings Saved",

                $"Recognition settings has been saved.",

                alertType: OFS.Lib.Utils.AlertType.Success);

        }

        private List<Camera> GetAllConnectedCamera()

        { var cameraNames = new List<Camera>();

                                        using (var searcher = new
ManagementObjectSearcher("SELECT * FROM Win32_PnPEntity
WHERE (PNPClass = 'Image' OR PNPClass = 'Camera')"))

            { var count = 0;

              foreach (var device in searcher.Get())

              {
```

```
                    cameraNames.Add(new Camera() { Name =
device["Caption"].ToString(), Index = count++ });
            }
        } return cameraNames;
    }
    private void trackBarControl1_EditValueChanged(object sender,
EventArgs e)
            var value = trackBarControl1.Value;
        lblCofidenceValue.Text = value.ToString();
    }
      private void trackBarTimeInInterval_EditValueChanged(object
sender, EventArgs e)
    {
        var value = trackBarTimeInInterval.Value;
        lblTimeIninterval.Text = value.ToString();
    }
      private void trackBarreadingSpeed_EditValueChanged(object
sender, EventArgs e)
    {
        var value = trackBarreadingSpeed.Value;
        lblReadingSpeed.Text = value.ToString();
    }
  }
}
```

APPENDIX H: CURRICULUM VITAE

CRISTINE JOY Q. SAGADRACA
ZONE 5 CUMABAO, TUMAUINI, ISABELA
(+63) 9616879262
its.me.cristine404@gmail.com

CAREER OBJECTIVE

A flexible and determined IT professional seeking to leverage my skills in creating efficient solutions and enhancing user experience, while continuously expanding my knowledge in the technology field.

EDUCATIONAL BACKGROUND

TERTIARY

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY
HGBaquiran College, Inc.
Tumauini, Isabela Philippines
2022 – 2026

SECONDARY

SENIOR HIGH SCHOOL

SCIENCE TECHNOLOGY ENGINEERING AND MATHEMATICS (STEM)
Advance Montessori Education Center of Isabela, Inc.
Barangay Maligaya, Tumauini, Isabela
2020 – 2022

JUNIOR HIGH SCHOOL

Advance Montessori Education Center of Isabela, Inc.
Barangay Maligaya, Tumauini, Isabela
2016 – 2020

PRIMARY

Come and See Christian School
Barangay Cumabao, Tumauini, Isabela
2010 – 2016

COMPUTING SKILLS

Software

Programming Languages

- C#, VB.Net, PHP, JavaScript

Database Engines

- MySQL, MS SQL Server

Web Development Tools/Framework

- HTML and CSS

Hardware

- Assembly And Dis-Assembly
- PC Troubleshooting

CAPSTONE PROJECT

- “Smart Campus Security: Facial Recognition-Based Entry And Attendance Monitoring System”

TRAINING, SEMINARS AND WEBINAR ATTENDED

- **Simulation with Real HR Professional : Bridging Classroom Learning to Real-World Readiness**
  Online Via Zoom
  November 2025

- **Regional Information Technology And Engineering Convention 2024**
  St. Paul University Philippines
  Tuguegarao City, Cagayan
  March 2024

DIAZ, JESUS DOMINGO
ZONE 1 ANTAGAN 2ND TUMAUINI, ISABELA
(+63) 948-909-5214
jesus.social@proton.me

CAREER OBJECTIVE

Life presents uncertainty and challenges, but I remain adaptable and resilient. I am committed to applying my IT skills to develop solutions that improve efficiency and enhance user experience, while continuously expanding my knowledge and expertise in the field.

EDUCATIONAL BACKGROUND

TERTIARY

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY
HGBaquiran College, Inc.
Tumauini, Isabela Philippines
2022 – 2026

SECONDARY

SENIOR HIGH SCHOOL

SCIENCE TECHNOLOGY ENGINEERING AND MATHEMATICS (STEM)
Advance Montessori Education Center of Isabela, Inc.
Barangay Maligaya, Tumauini, Isabela
2020 – 2022

JUNIOR HIGH SCHOOL
Advance Montessori Education Center of Isabela, Inc.
Barangay Maligaya, Tumauini, Isabela

2016 – 2020

PRIMARY

Advance Montessori Education Center of Isabela, Inc.
Barangay Maligaya, Tumauini, Isabela
2010 – 2016

COMPUTING SKILLS

Software

Programming Languages

- C#, VB.Net, PHP, JavaScript

Database Engines

- MySQL, MS SQL Server

Web Development Tools/Framework

- HTML, CSS, and Closed/Open API'S

Hardware

- Assembly And Dis-Assembly
- PC Troubleshooting

CAPSTONE PROJECT

- "Smart Campus Security: Facial Recognition-Based Entry And Attendance Monitoring System"

TRAINING, SEMINARS AND WEBINAR ATTENDED

- **Regional Information Technology And Engineering Convention 2024**
  St. Paul University Philippines
  Tuguegarao City, Cagayan
  March 2024

CHELLSY FEI B. MANALO
BARANGAY LANNA TUMAUINI, ISABELA
(+63) 928-493-6025
manalochellysyfei@gmail.com

CAREER OBJECTIVE

Motivated IT girl seeking an entry-level role to apply technical and problem-solving skills while growing professionally and contributing to organizational success.

EDUCATIONAL BACKGROUND

TERTIARY

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY
HGBaquiran College, Inc.
Tumauini, Isabela Philippines
2022 – 2026

SECONDARY

SENIOR HIGH SCHOOL

GENERAL ACADEMIC STRAND (GAS)
Lanna National High School
Barangay Lanna, Tumauini, Isabela
2020 – 2022

JUNIOR HIGH SCHOOL

Lanna National High School
Barangay Lanna, Tumauini, Isabela
2016 – 2020

PRIMARY

Lanna Elementary School
Barangay Lanna, Tumauini, Isabela
2010 – 2016

COMPUTING SKILLS

Software

Programming Languages

- C#, VB.Net, PHP, JavaScript

Web Development Tools/Framework

- HTML, CSS, and Closed/Open API'S

Hardware

- PC Troubleshooting

CAPSTONE PROJECT

- "Smart Campus Security: Facial Recognition-Based Entry And Attendance Monitoring System"

TRAINING, SEMINARS AND WEBINAR ATTENDED

- **Regional Information Technology And Engineering Convention 2024**
  St. Paul University Philippines
  Tuguegarao City, Cagayan
  March 2024

ISENLOUIS L. ONG
BARANGAY IV TUMAUINI, ISABELA
(+63) 9614146046
isenong76@gmail.com

CAREER OBJECTIVE

An adaptable and innovative individual with computer servicing and communication skills. I strive to apply these competencies to drive organizational success, deliver practical solutions, and contribute meaningful value in dynamic and collaborative environments.

EDUCATIONAL BACKGROUND

TERTIARY

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY
HGBaquiran College, Inc.
Tumauini, Isabela Philippines
2020 – 2026

SECONDARY

SENIOR HIGH SCHOOL

COMPUTER SYSTEMS SERVICING (TVL)
Tumauini National Highschool
Barangay Annafunan, Tumauini, Isabela
2018 – 2020

JUNIOR HIGH SCHOOL
Tumauini National Highschool
Barangay Annafunan, Tumauini, Isabela
2014 – 2017

PRIMARY

Tumauini North Central School
Barangay San Pedro, Tumauini, Isabela
2008 – 2013

COMPUTING SKILLS

Software

Programming Languages

- C#

Web Development Tools/Framework

- HTML, CSS, and Closed/Open API'S

Hardware

- Assembly And Dis-Assembly
- PC Troubleshooting

CAPSTONE PROJECT

- "Smart Campus Security: Facial Recognition-Based Entry And Attendance Monitoring System"

TRAINING, SEMINARS AND WEBINAR ATTENDED

- **Regional Information Technology And Engineering Convention 2024**
  St. Paul University Philippines
  Tuguegarao City, Cagayan
  March 2024